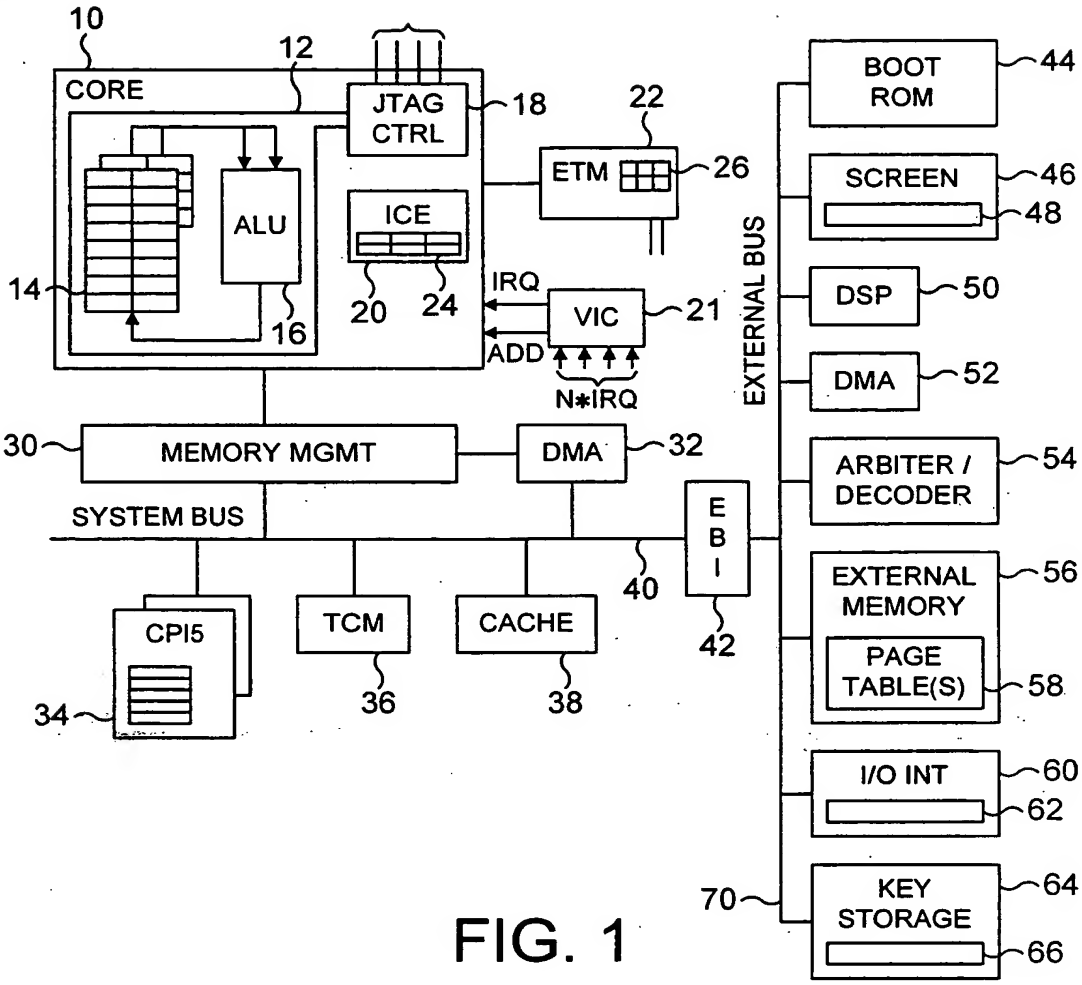

FIG. 1

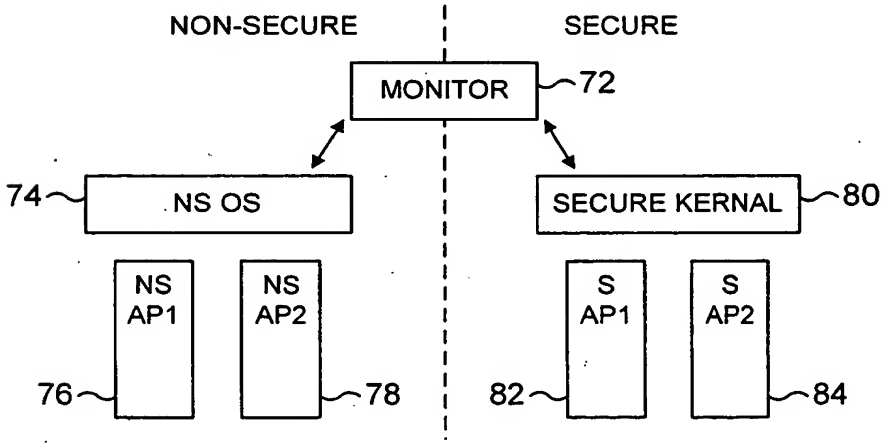

FIG. 2

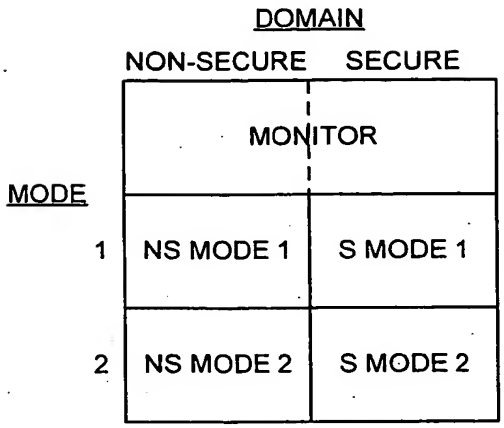

FIG. 3

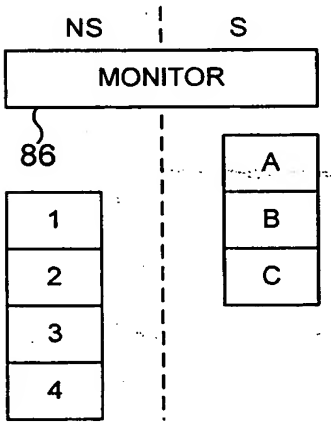

FIG. 4

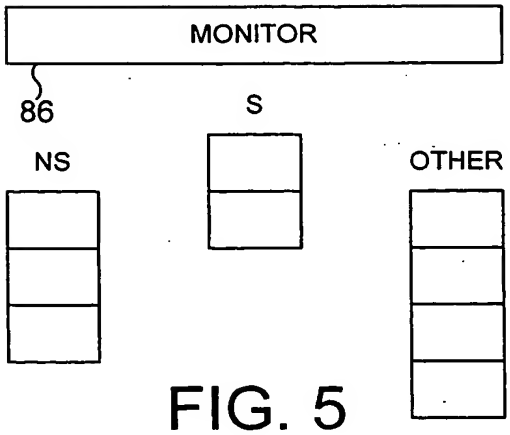

FIG. 5

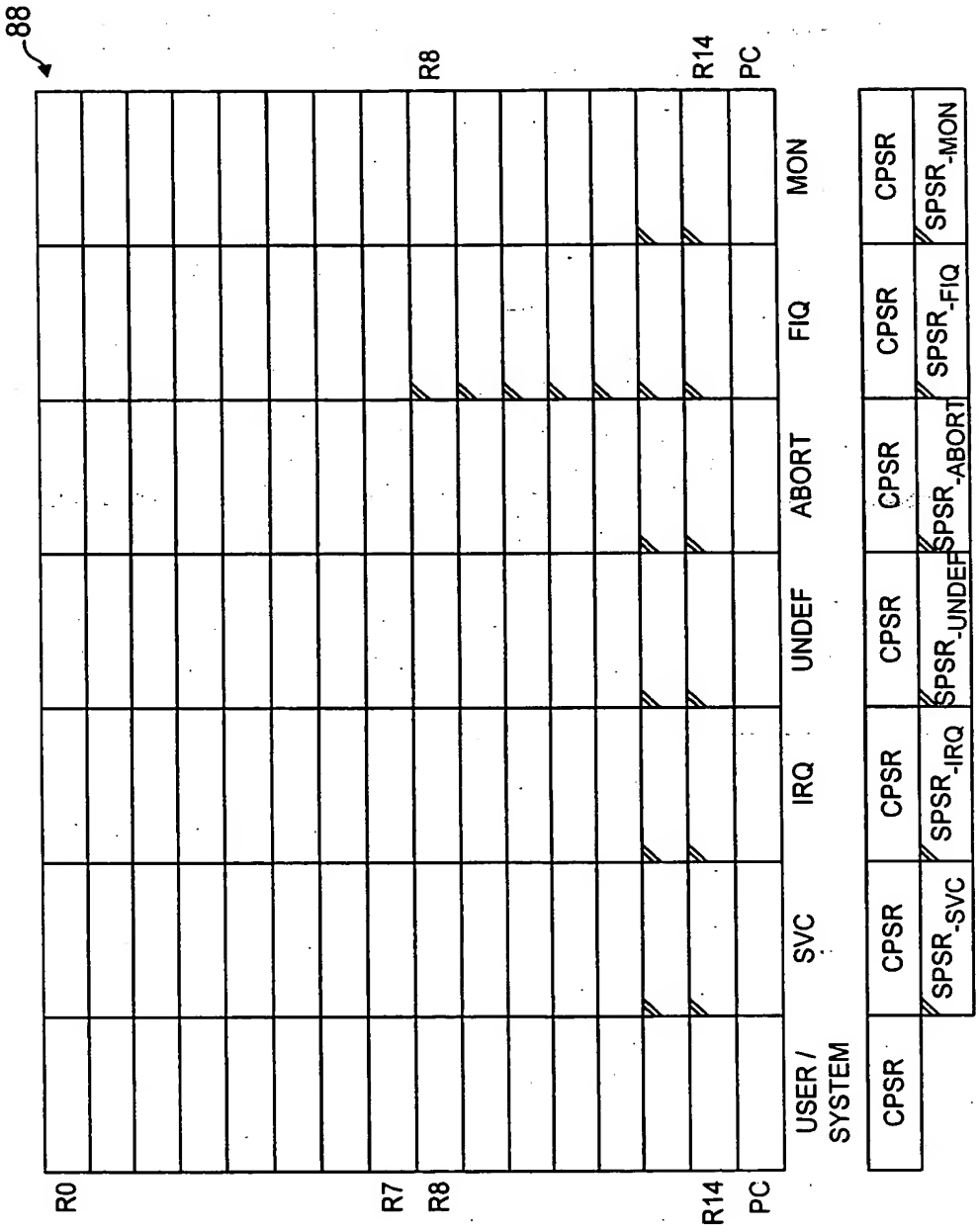
88
R0
R7
R8
R8
R14
R14
PC
PC
USER / SYSTEM
SVC
IRQ
UNDEF
ABORT
FIQ
MON
CPSR
CPSR
SPSR_SVC
CPSR
SPSR_IRQ
CPSR
SPSR_UNDEF
CPSR
SPSR_ABORT
CPSR
SPSR_FIQ
CPSR
SPSR_MON
// = PRIVATE TO MODE

FIG. 6

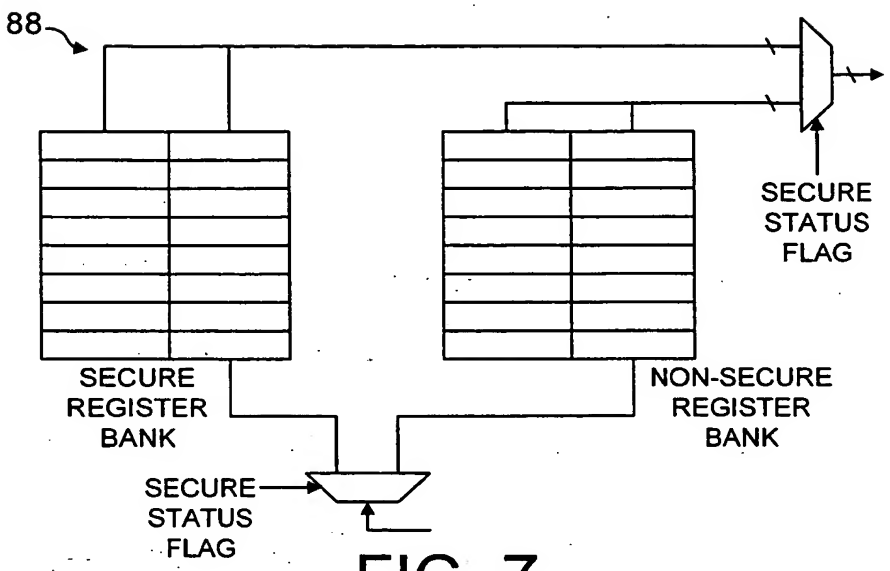
88
SECURE
STATUS
FLAG
SECURE
REGISTER
BANK
NON-SECURE
REGISTER
BANK
SECURE
STATUS
FLAG

FIG. 7

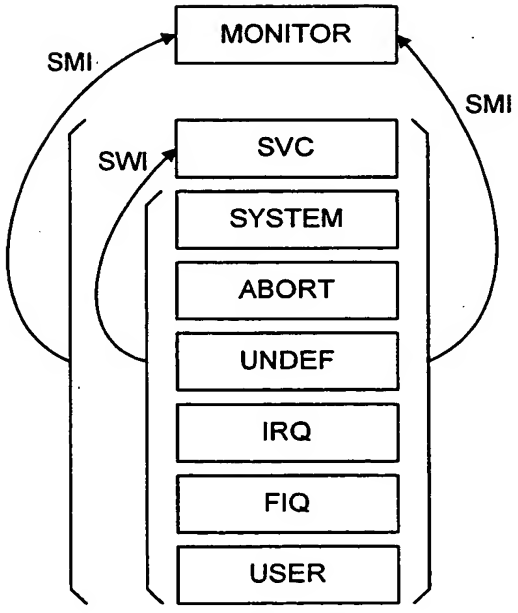
MONITOR
SMI
SMI
SWI
SVC
SYSTEM
ABORT
UNDEF
IRQ
FIQ
USER

FIG. 8

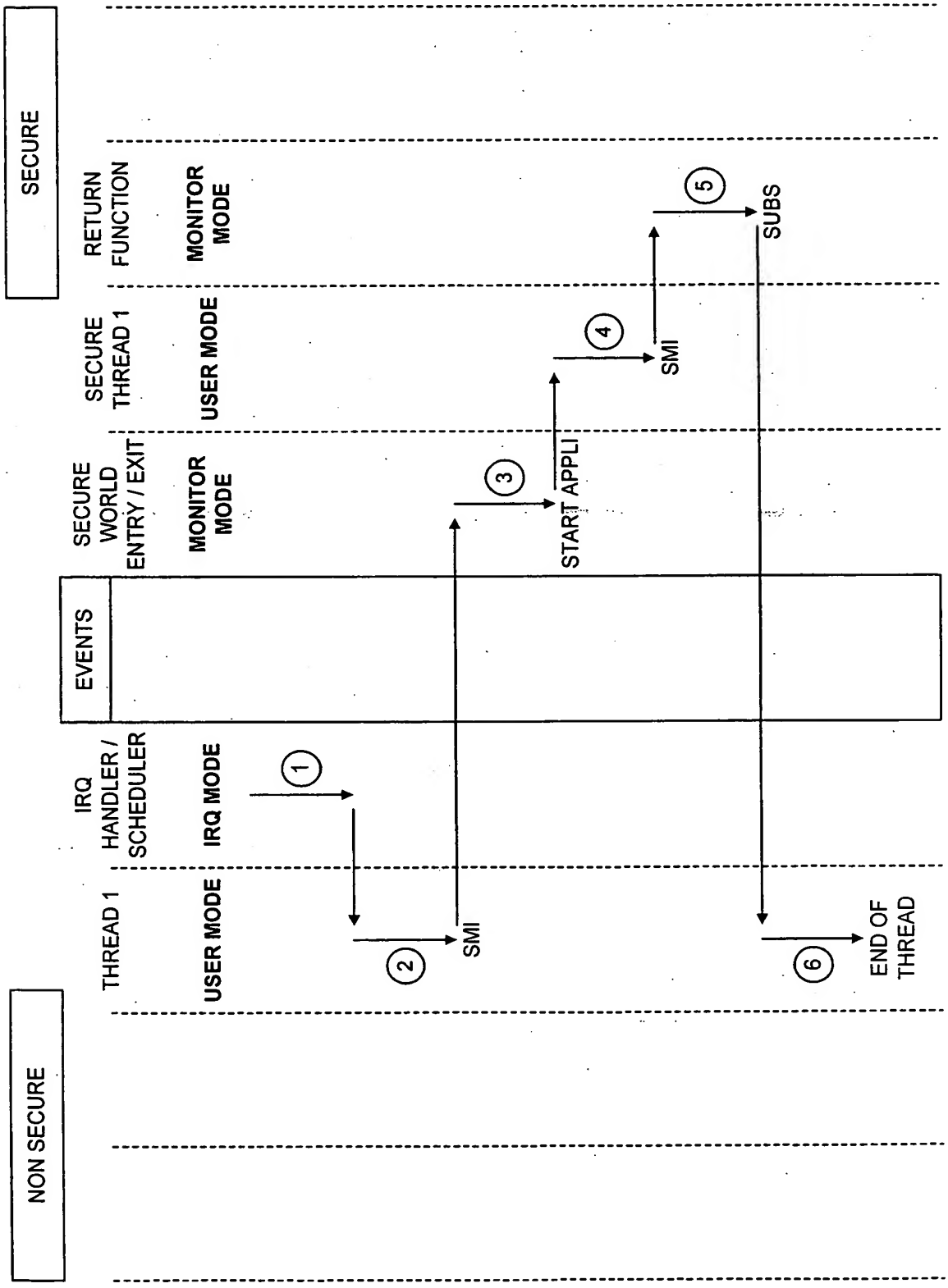
SECURE
NON SECURE
THREAD 1
USER MODE
IRQ HANDLER / SCHEDULER
IRQ MODE
SECURE WORLD ENTRY / EXIT
MONITOR MODE
SECURE THREAD 1
USER MODE
RETURN FUNCTION
MONITOR MODE
EVENTS
1
2
SMI
3
START APPLI
4
SMI
5
SUBS
6
END OF THREAD

FIG. 9

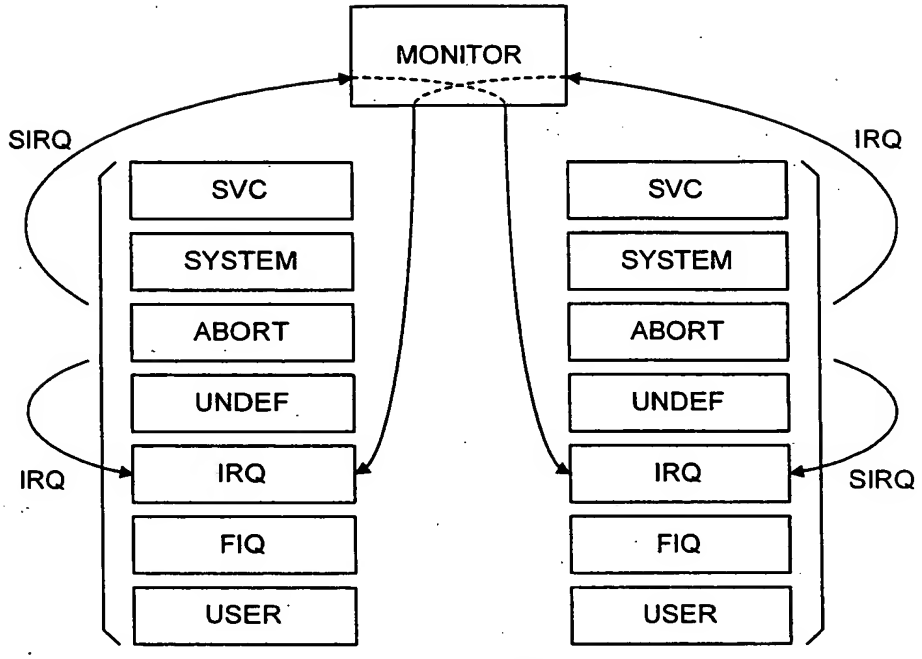
MONITOR
SIRQ
IRQ
SVC
SYSTEM
ABORT
UNDEF
IRQ
FIQ
USER
SVC
SYSTEM
ABORT
UNDEF
IRQ
FIQ
USER
IRQ
SIRQ

FIG. 10

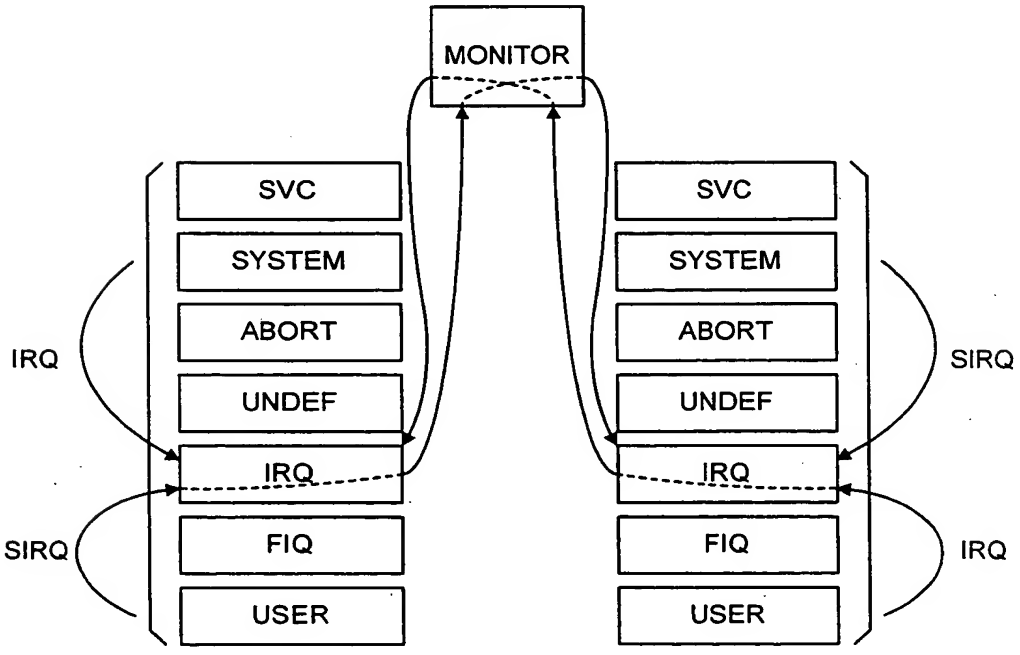
MONITOR
SVC
SYSTEM
ABORT
UNDEF
IRQ
FIQ
USER
SVC
SYSTEM
ABORT
UNDEF
IRQ
FIQ
USER
IRQ
SIRQ
SIRQ
IRQ

FIG. 12

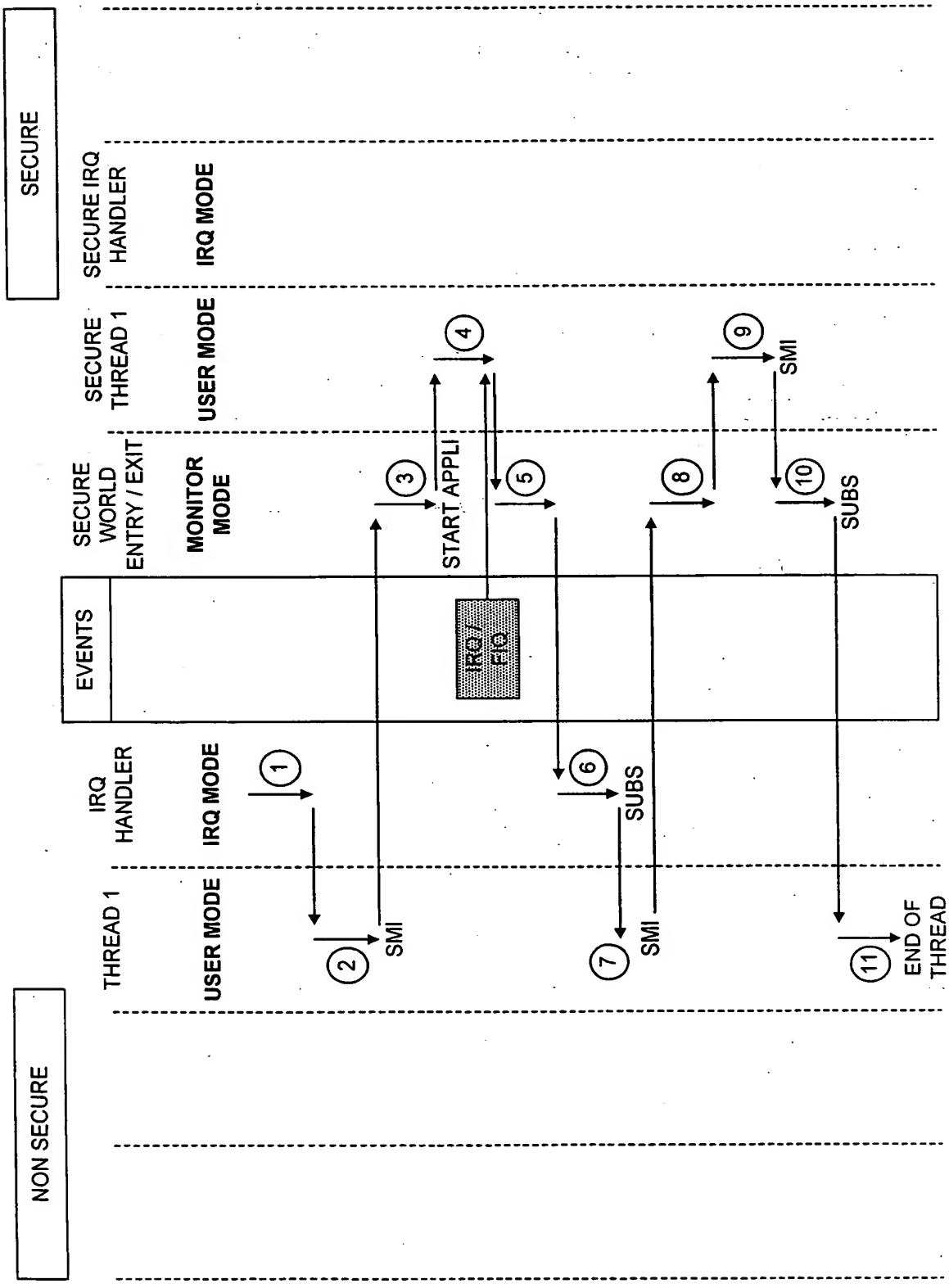
SECURE
NON SECURE
SECURE IRQ HANDLER
IRQ MODE
SECURE THREAD 1
USER MODE
SECURE WORLD ENTRY / EXIT
MONITOR MODE
EVENTS
IRQ / FIQ
IRQ HANDLER
IRQ MODE
THREAD 1
USER MODE
START APPLI
SUBS
SMI
END OF THREAD
1
2
3
4
5
6
7
8
9
10
11

FIG. 11A

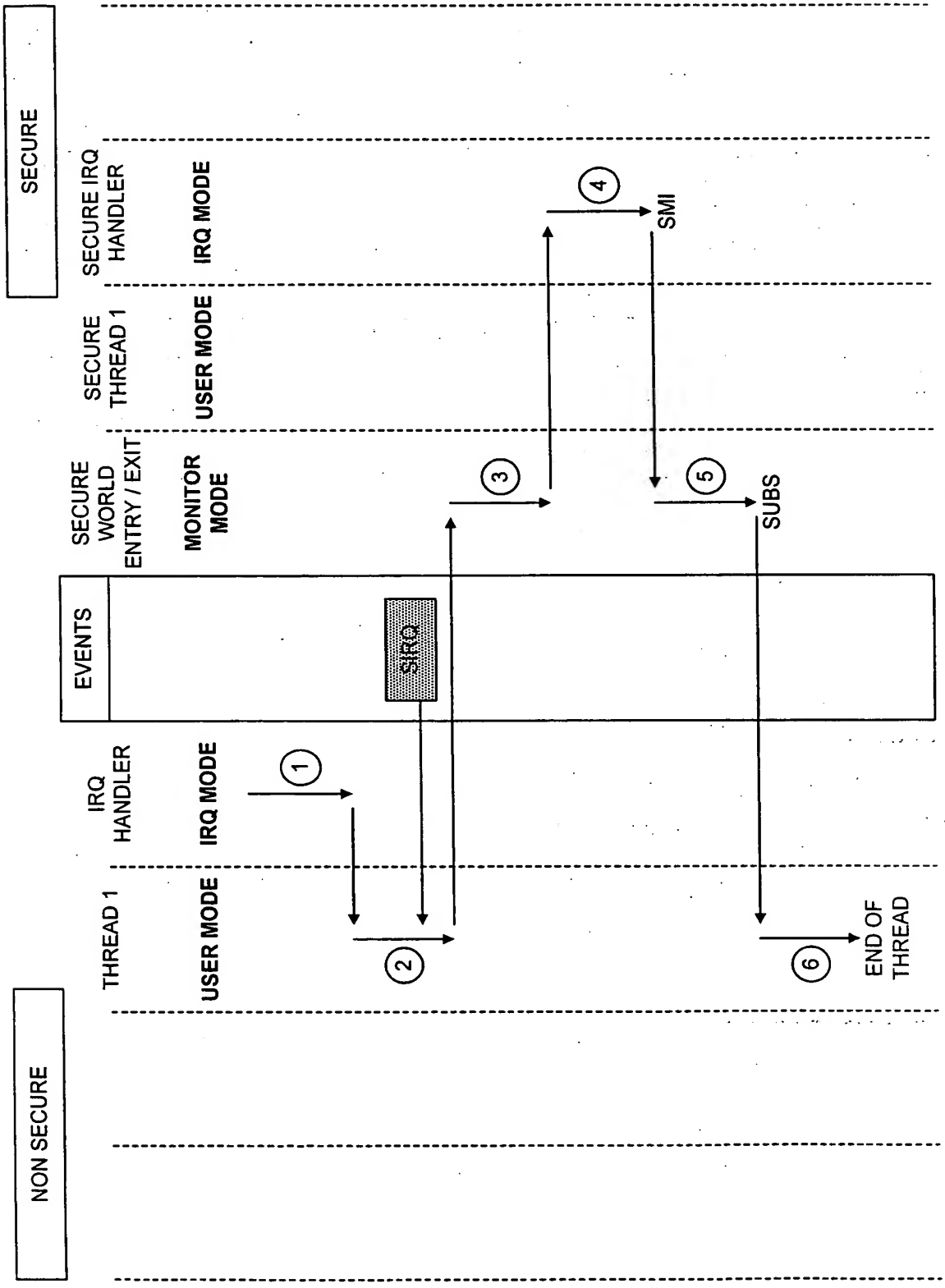
SECURE
NON SECURE
SECURE IRQ HANDLER
IRQ MODE
SECURE THREAD 1
USER MODE
SECURE WORLD ENTRY / EXIT
MONITOR MODE
EVENTS
IRQ HANDLER
IRQ MODE
THREAD 1
USER MODE
SIRQ
SMI
SUBS
END OF THREAD
1
2
3
4
5
6

FIG. 11B

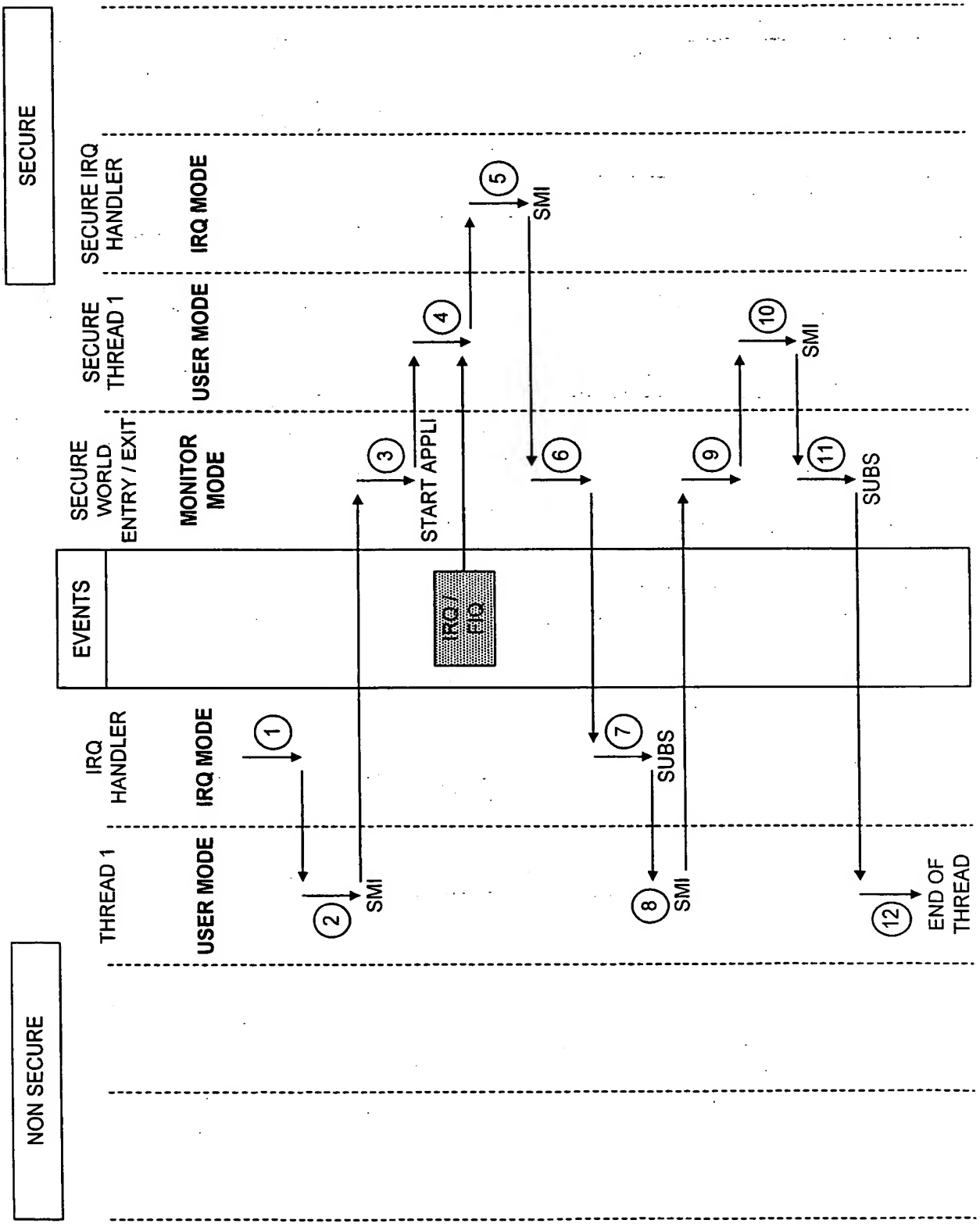
SECURE
NON SECURE
THREAD 1
USER MODE
IRQ HANDLER
IRQ MODE
SECURE THREAD 1
USER MODE
SECURE IRQ HANDLER
IRQ MODE
SECURE WORLD ENTRY / EXIT
MONITOR MODE
EVENTS
IRQ / FIQ
1
2
SMI
3
START APPLI
4
5
SMI
6
7
SUBS
8
SMI
9
10
SMI
11
SUBS
12
END OF THREAD

FIG. 13A

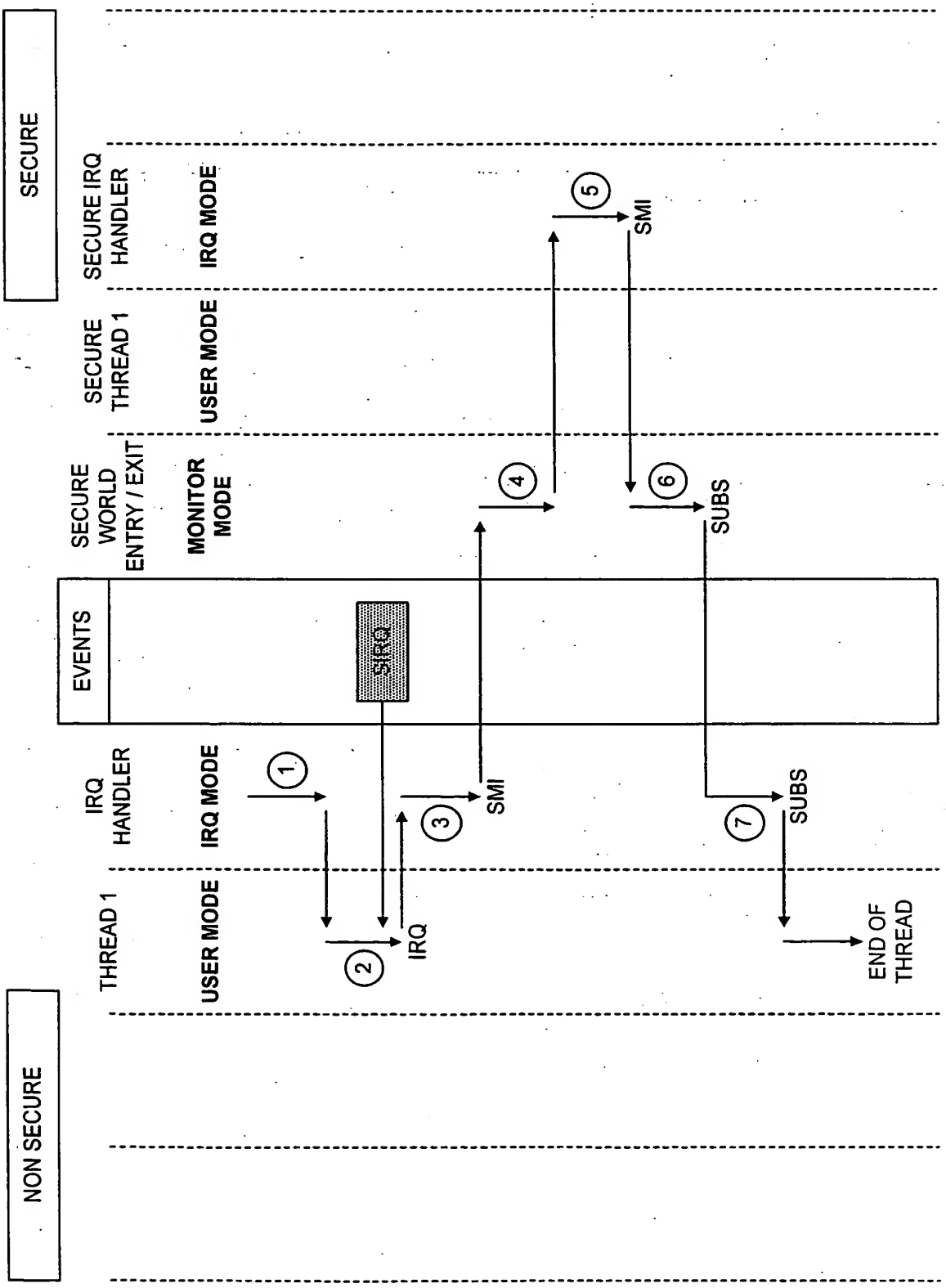
SECURE
NON SECURE
SECURE IRQ HANDLER
IRQ MODE
SECURE THREAD 1
USER MODE
SECURE WORLD ENTRY / EXIT
MONITOR MODE
EVENTS
SIRQ
IRQ HANDLER
IRQ MODE
THREAD 1
USER MODE
IRQ
SMI
SMI
SUBS
SUBS
END OF THREAD
1
2
3
4
5
6
7

FIG. 13B

EXCEPTION	VECTOR OFFSET	CORRESPONDING MODE
RESET	0x00	SUPERVISOR MODE
UNDEF	0x04	MONITOR MODE / UNDEF MODE
SWI	0x08	SUPERVISOR MODE / MONITOR MODE
PREFETCH ABORT	0x0C	ABORT MODE / MONITOR MODE
DATA ABORT	0x10	ABORT MODE / MONITOR MODE
IRQ / SIRQ	0x18	IRQ MODE / MONITOR MODE
FIQ	0x1C	FIQ MODE / MONITOR MODE
SMI	0x20	UNDEF MODE / MONITOR MODE

FIG. 14

MONITOR

RESET	VM0
UNDEF	VM1
SWI	VM2
PREFETCH ABORT	VM3
DATA ABORT	VM4
IRQ / SIRQ	VM5
FIQ	VM6
SMI	VM7

SECURE

RESET	VS0
UNDEF	VS1
SWI	VS2
PREFETCH ABORT	VS3
DATA ABORT	VS4
IRQ / SIRQ	VS5
FIQ	VS6
SMI	VS7

NON-SECURE

RESET	VNS0
UNDEF	VNS1
SWI	VNS2
PREFETCH ABORT	VNS3
DATA ABORT	VNS4
IRQ / SIRQ	VNS5
FIQ	VNS6
SMI	VNS7

FIG. 15

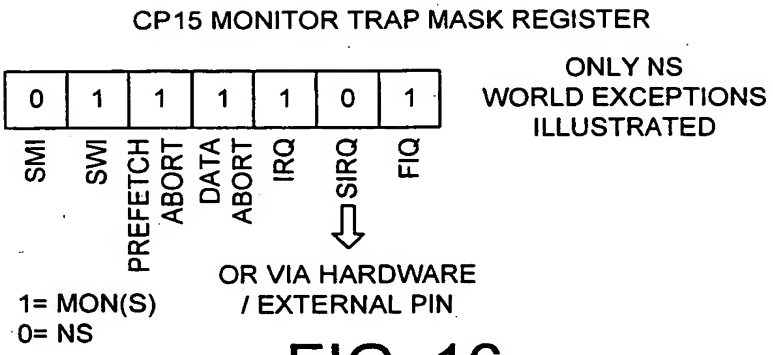
CP15 MONITOR TRAP MASK REGISTER
0
1
1
1
1
0
1
ONLY NS
WORLD EXCEPTIONS
ILLUSTRATED
SMI
SWI
PREFETCH ABORT
DATA ABORT
IRQ
SIRQ
FIQ
OR VIA HARDWARE
/ EXTERNAL PIN
1= MON(S)
0= NS

FIG. 16

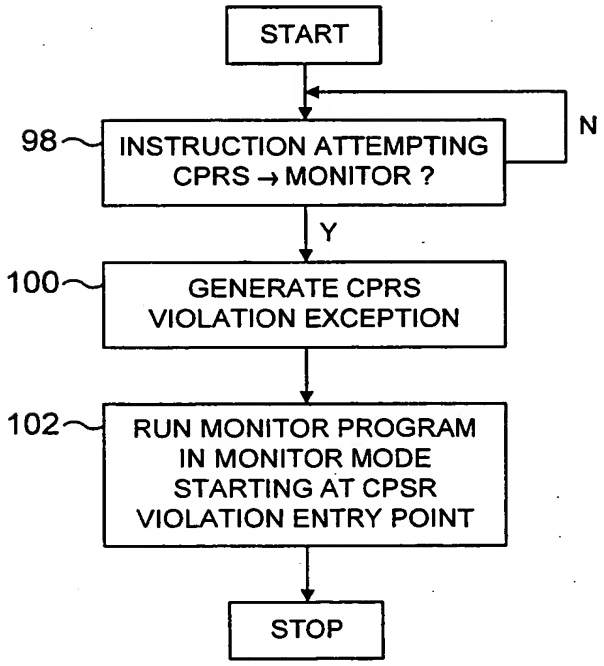
START
98
INSTRUCTION ATTEMPTING
CPRS → MONITOR ?
N
Y
100
GENERATE CPRS
VIOLATION EXCEPTION
102
RUN MONITOR PROGRAM
IN MONITOR MODE
STARTING AT CPSR
VIOLATION ENTRY POINT
STOP

FIG. 17

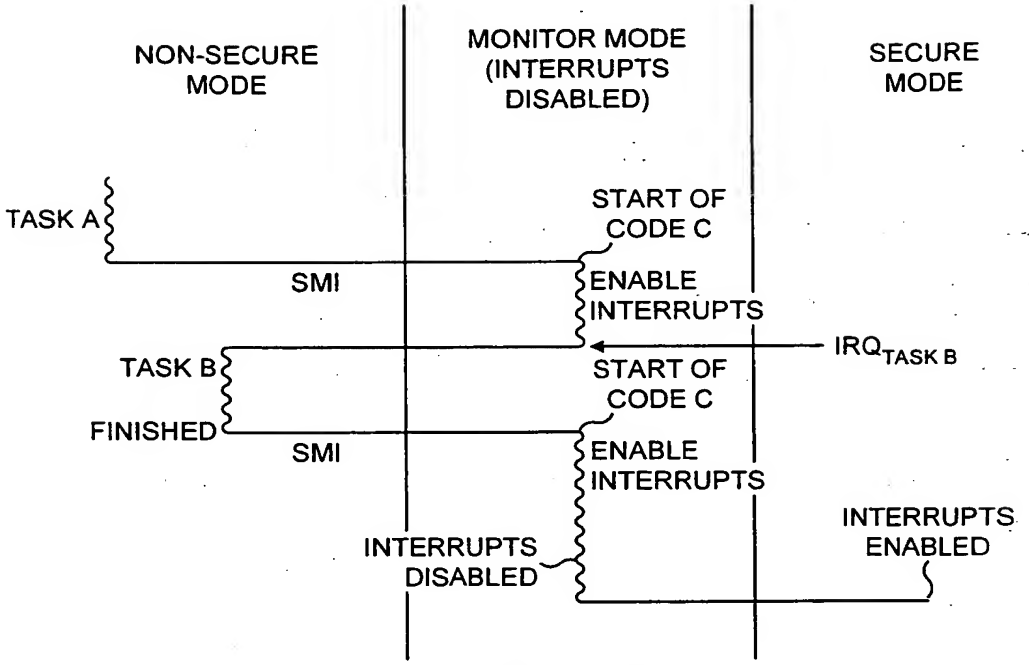
NON-SECURE MODE
MONITOR MODE (INTERRUPTS DISABLED)
SECURE MODE
TASK A
START OF CODE C
SMI
ENABLE INTERRUPTS
IRQ TASK B
TASK B
FINISHED
START OF CODE C
SMI
ENABLE INTERRUPTS
INTERRUPTS ENABLED
INTERRUPTS DISABLED

FIG. 18

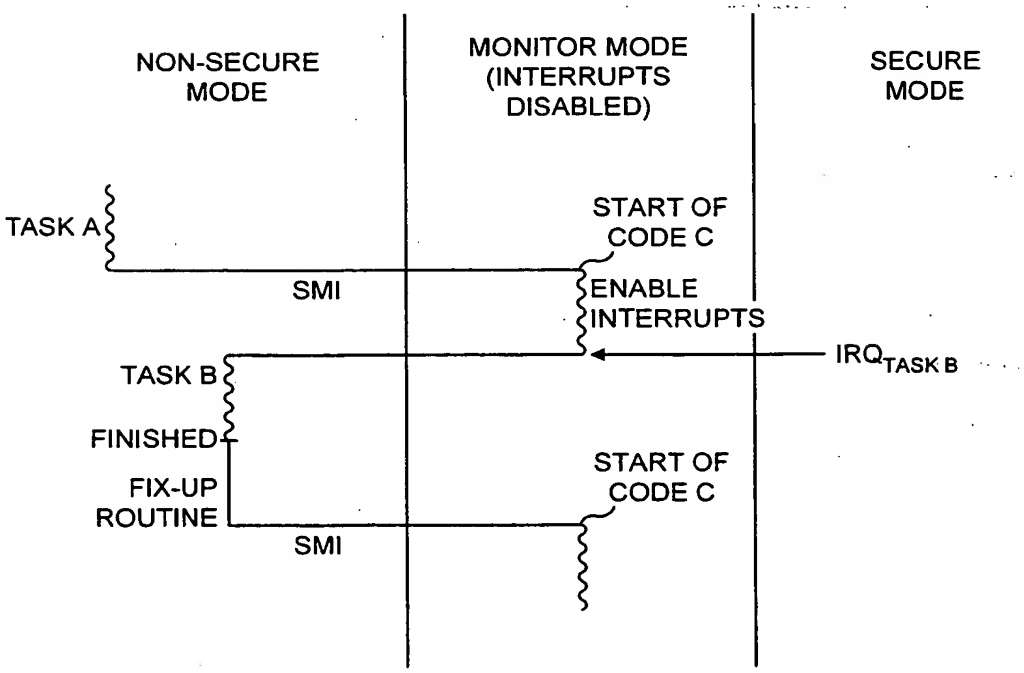
NON-SECURE MODE
MONITOR MODE (INTERRUPTS DISABLED)
SECURE MODE
TASK A
START OF CODE C
SMI
ENABLE INTERRUPTS
IRQ TASK B
TASK B
FINISHED
FIX-UP ROUTINE
START OF CODE C
SMI

FIG. 19

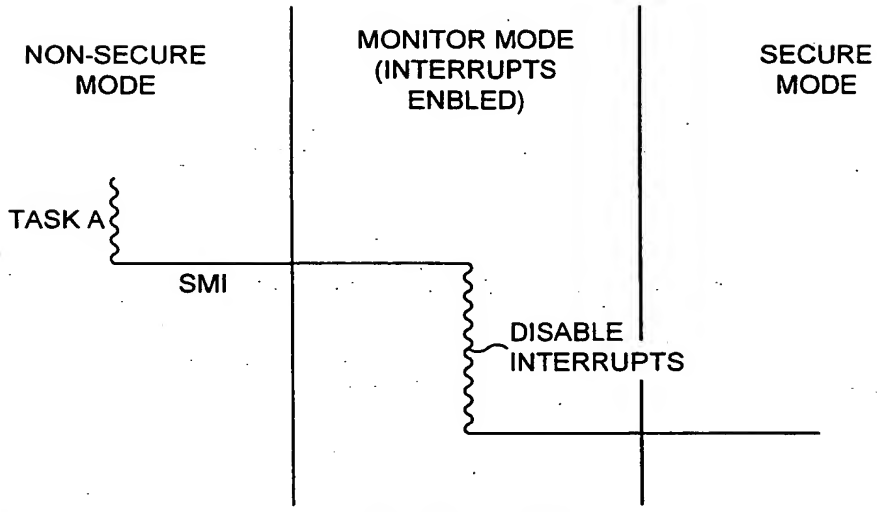
NON-SECURE MODE
MONITOR MODE (INTERRUPTS ENBLED)
SECURE MODE
TASK A
SMI
DISABLE INTERRUPTS

FIG. 20

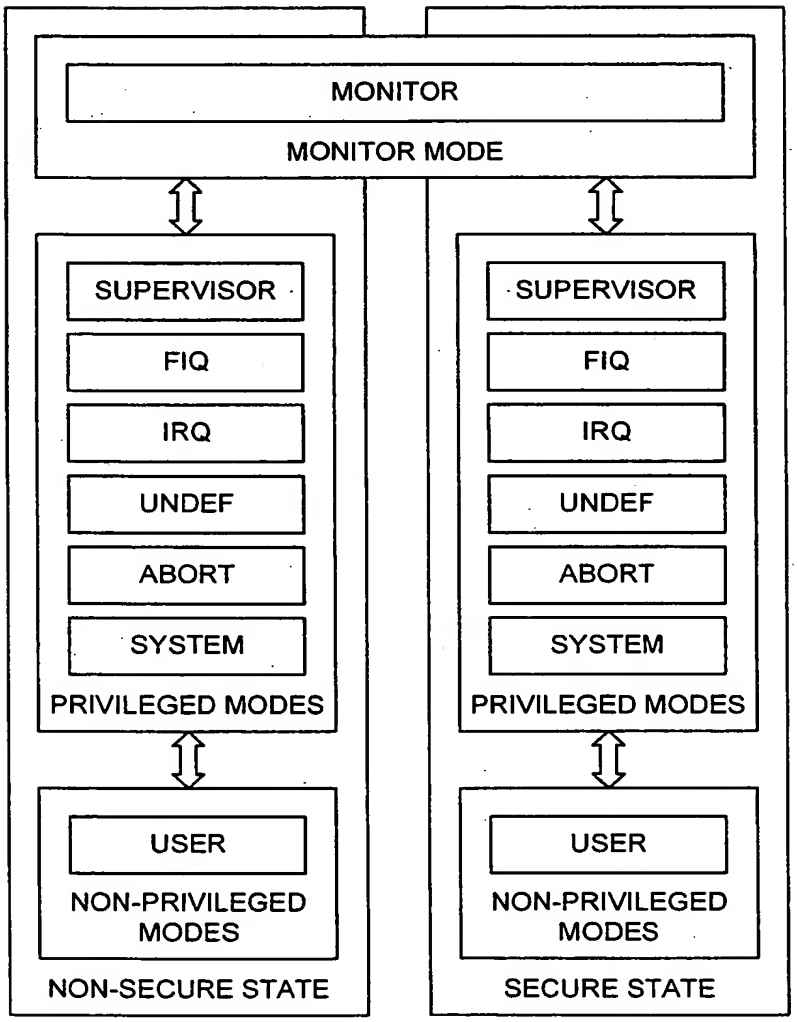
MONITOR
MONITOR MODE
SUPERVISOR
FIQ
IRQ
UNDEF
ABORT
SYSTEM
PRIVILEGED MODES
USER
NON-PRIVILEGED MODES
NON-SECURE STATE
SUPERVISOR
FIQ
IRQ
UNDEF
ABORT
SYSTEM
PRIVILEGED MODES
USER
NON-PRIVILEGED MODES
SECURE STATE

FIG. 21

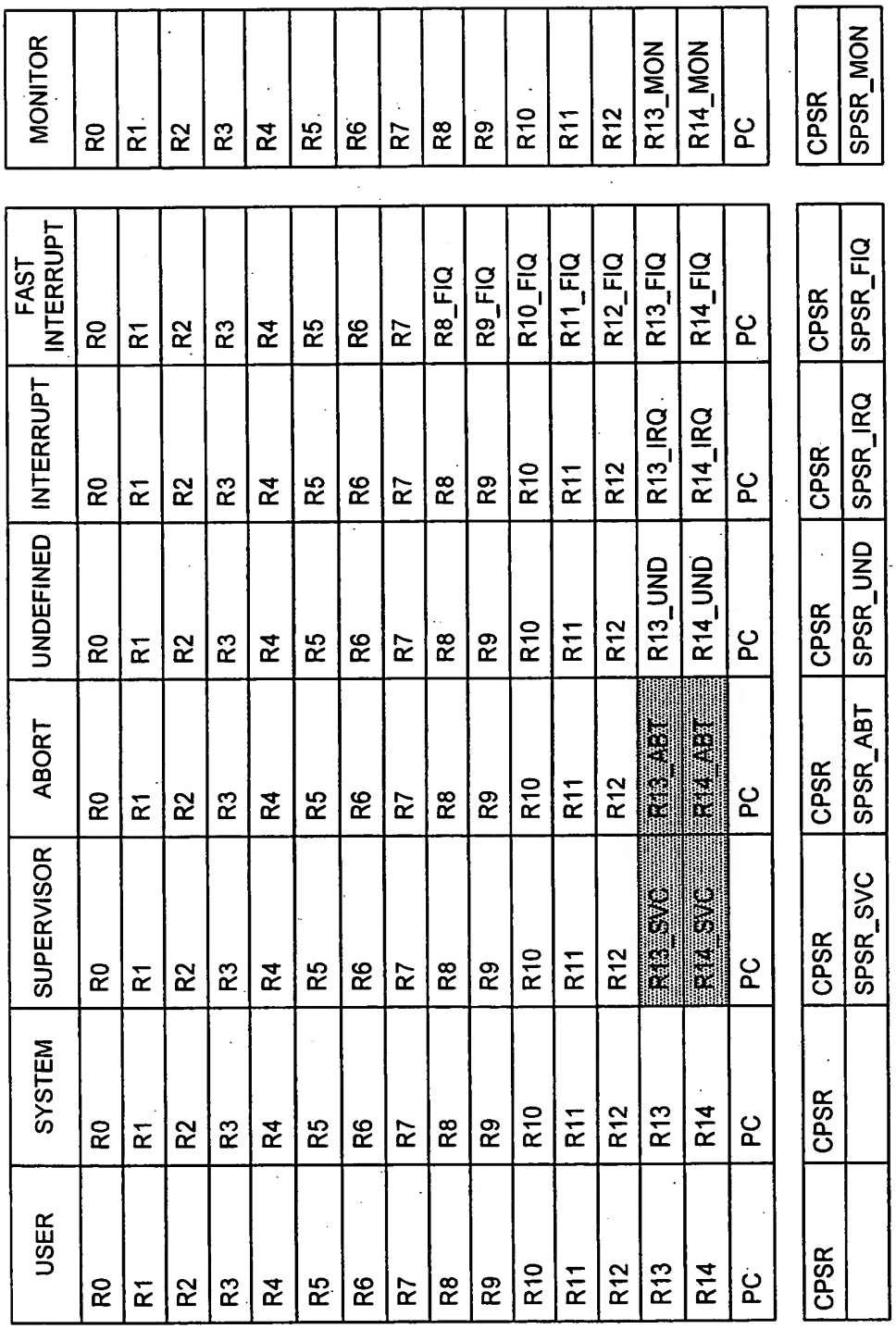
USER
SYSTEM
SUPERVISOR
ABORT
UNDEFINED
INTERRUPT
FAST INTERRUPT
MONITOR
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
PC
R13_SVC
R14_SVC
R13_ABT
R14_ABT
R13_UND
R14_UND
R13_IRQ
R14_IRQ
R8_FIQ
R9_FIQ
R10_FIQ
R11_FIQ
R12_FIQ
R13_FIQ
R14_FIQ
R13_MON
R14_MON
CPSR
SPSR_SVC
SPSR_ABT
SPSR_UND
SPSR_IRQ
SPSR_FIQ
SPSR_MON

FIG. 22

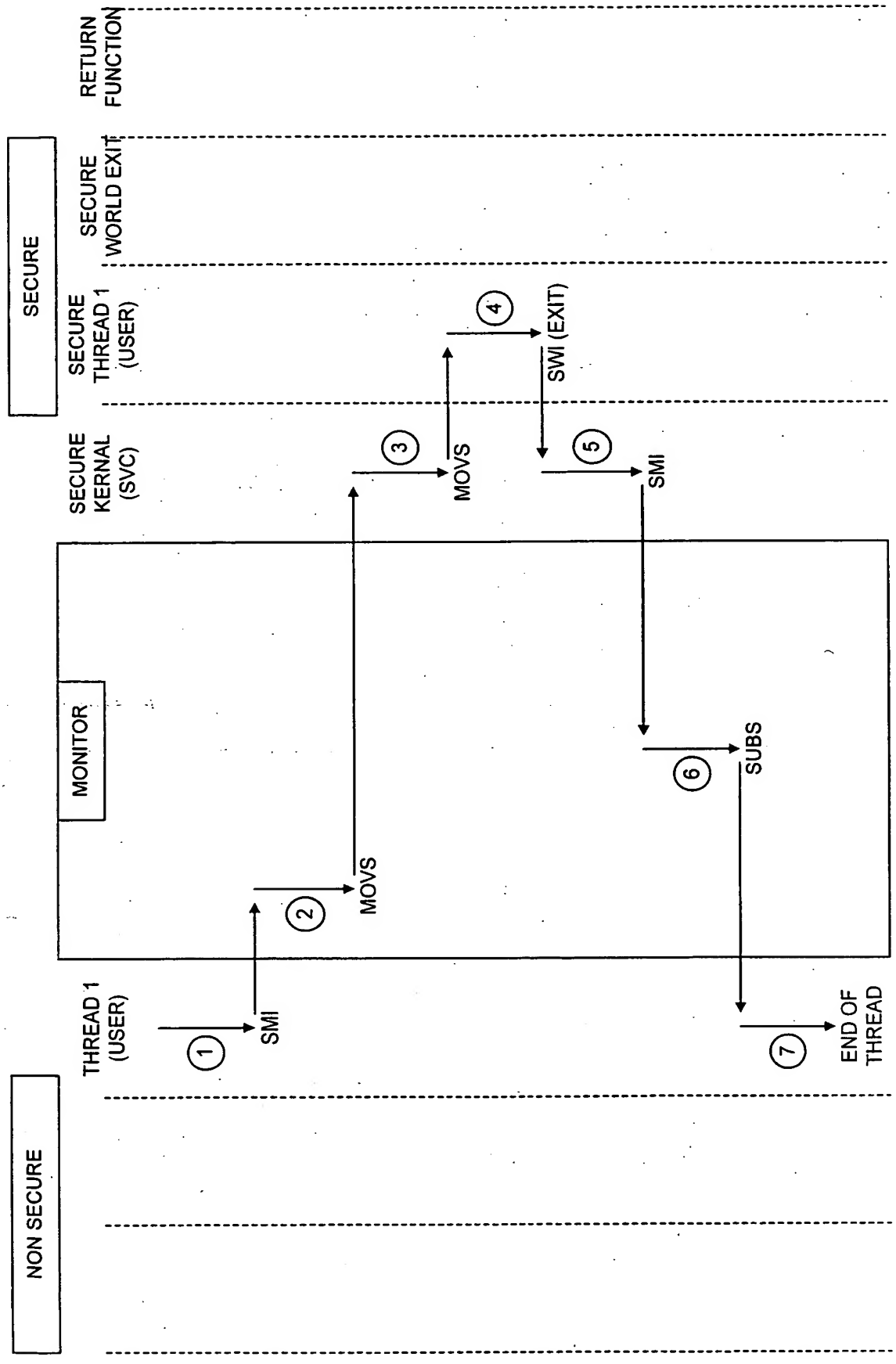
SECURE
NON SECURE
RETURN FUNCTION
SECURE WORLD EXIT
SECURE THREAD 1 (USER)
SECURE KERNAL (SVC)
MONITOR
THREAD 1 (USER)
SMI
MOVS
MOVS
SWI (EXIT)
SMI
SUBS
END OF THREAD
1
2
3
4
5
6
7

FIG. 23

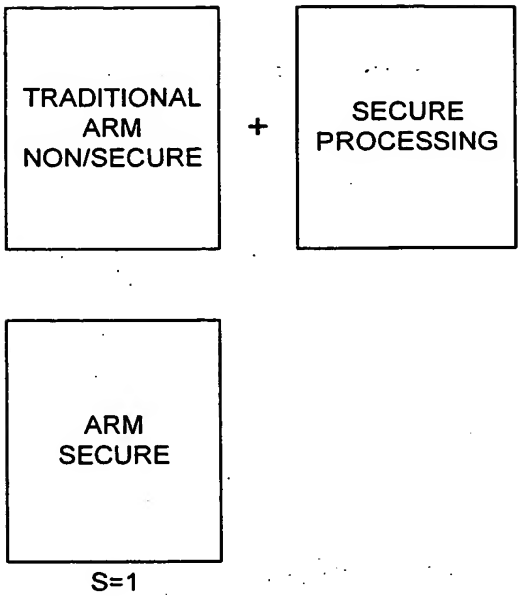
TRADITIONAL ARM NON/SECURE
+
SECURE PROCESSING
ARM SECURE
S=1

FIG. 24

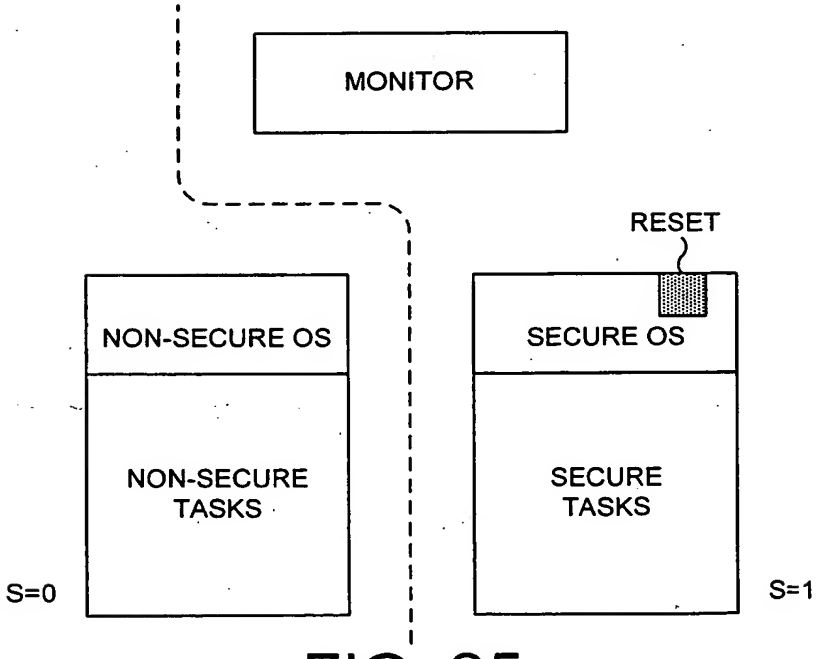
MONITOR
RESET
NON-SECURE OS
NON-SECURE TASKS
SECURE OS
SECURE TASKS
S=0
S=1

FIG. 25

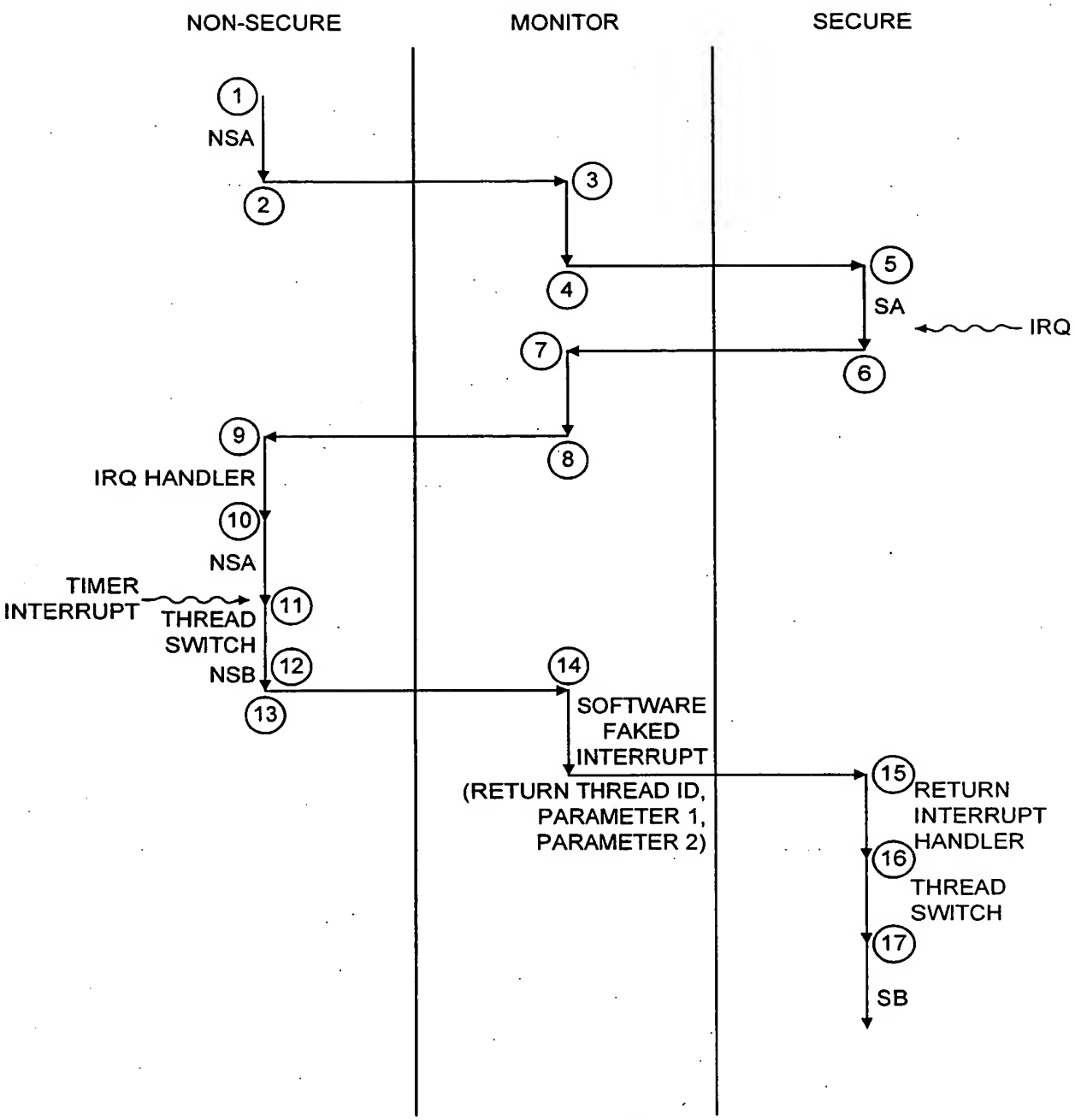
NON-SECURE
MONITOR
SECURE
1
NSA
2
3
4
5
SA
IRQ
6
7
8
9
IRQ HANDLER
10
NSA
TIMER INTERRUPT
THREAD SWITCH
11
NSB
12
13
14
SOFTWARE FAKED INTERRUPT
(RETURN THREAD ID, PARAMETER 1, PARAMETER 2)
15
RETURN INTERRUPT HANDLER
16
THREAD SWITCH
17
SB

FIG. 26

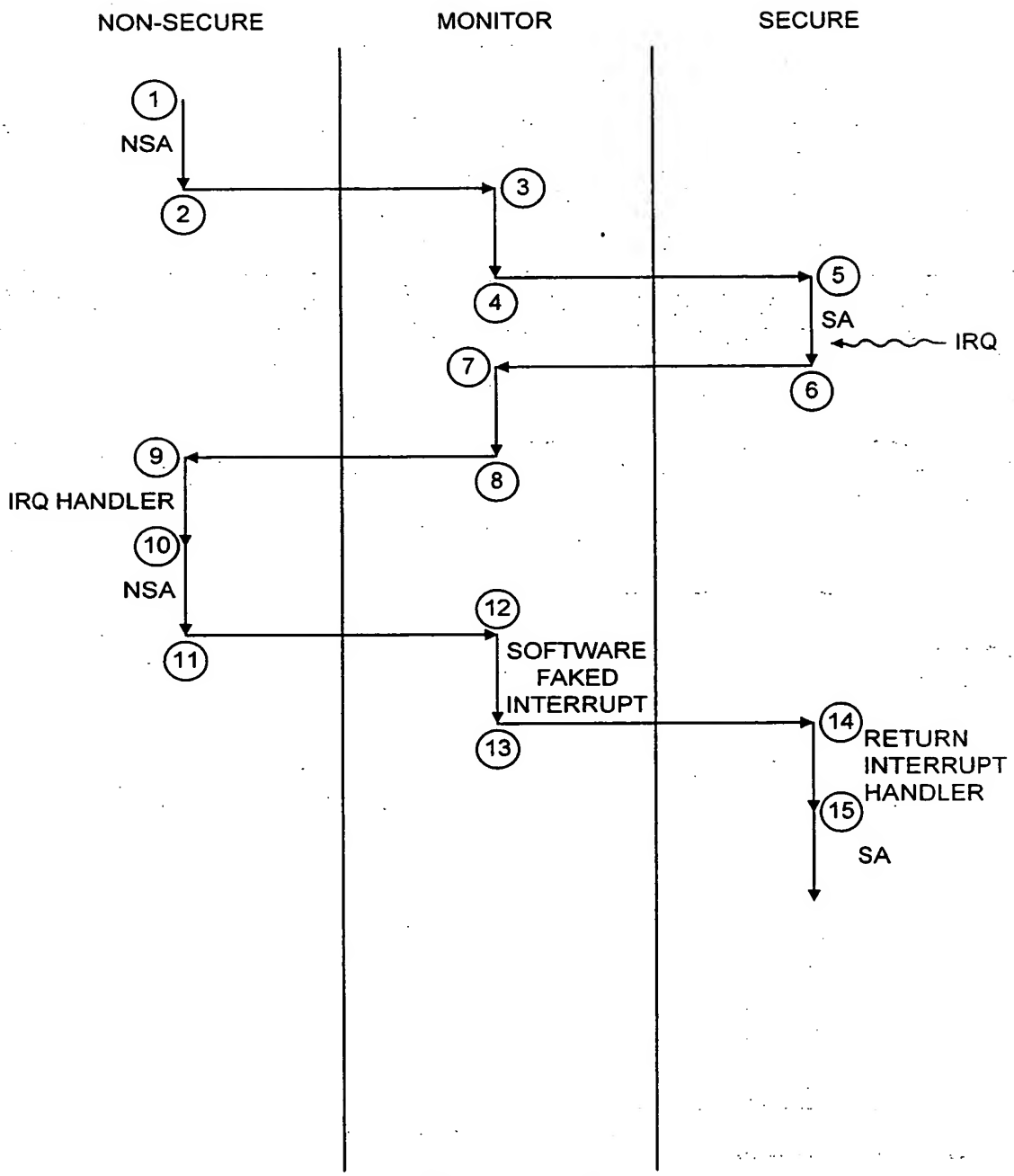
NON-SECURE
MONITOR
SECURE
1
NSA
2
3
4
5
SA
IRQ
6
7
8
9
IRQ HANDLER
10
NSA
11
12
SOFTWARE
FAKED
INTERRUPT
13
14
RETURN
INTERRUPT
HANDLER
15
SA

FIG. 27

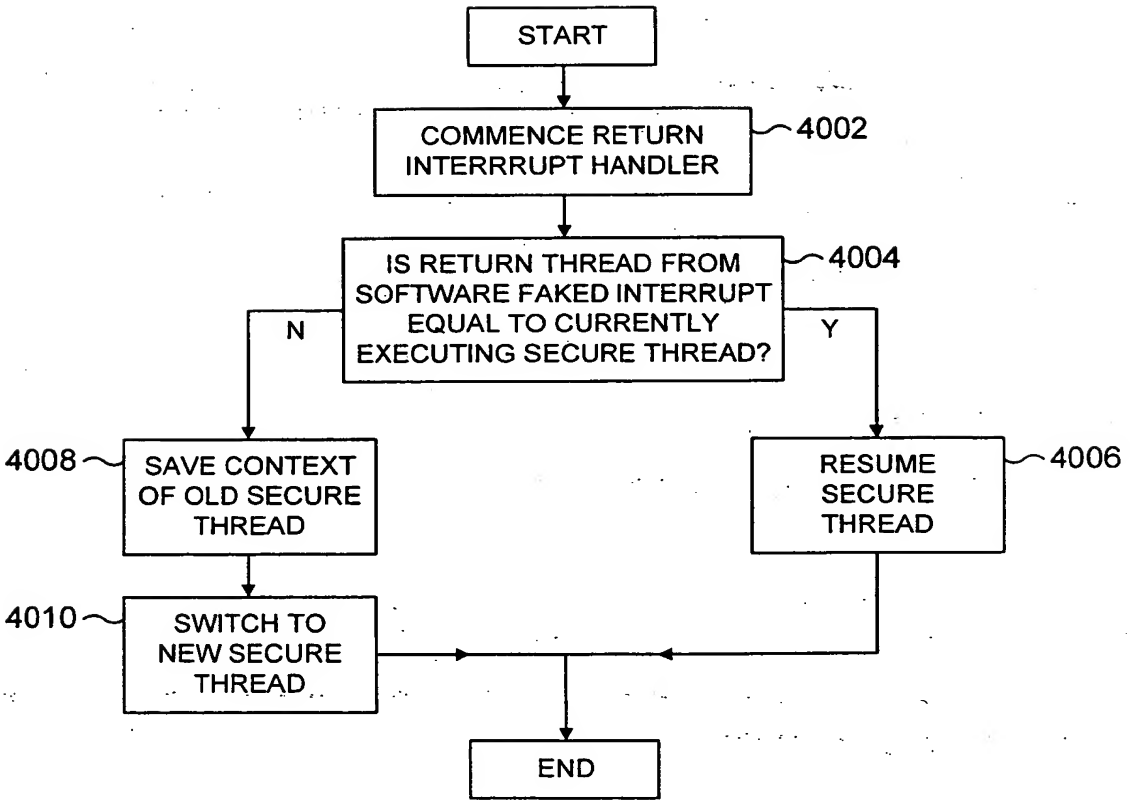
START
COMMENCE RETURN INTERRRUPT HANDLER
4002
IS RETURN THREAD FROM SOFTWARE FAKED INTERRUPT EQUAL TO CURRENTLY EXECUTING SECURE THREAD?
4004
N
Y
4008
SAVE CONTEXT OF OLD SECURE THREAD
RESUME SECURE THREAD
4006
4010
SWITCH TO NEW SECURE THREAD
END

FIG. 28

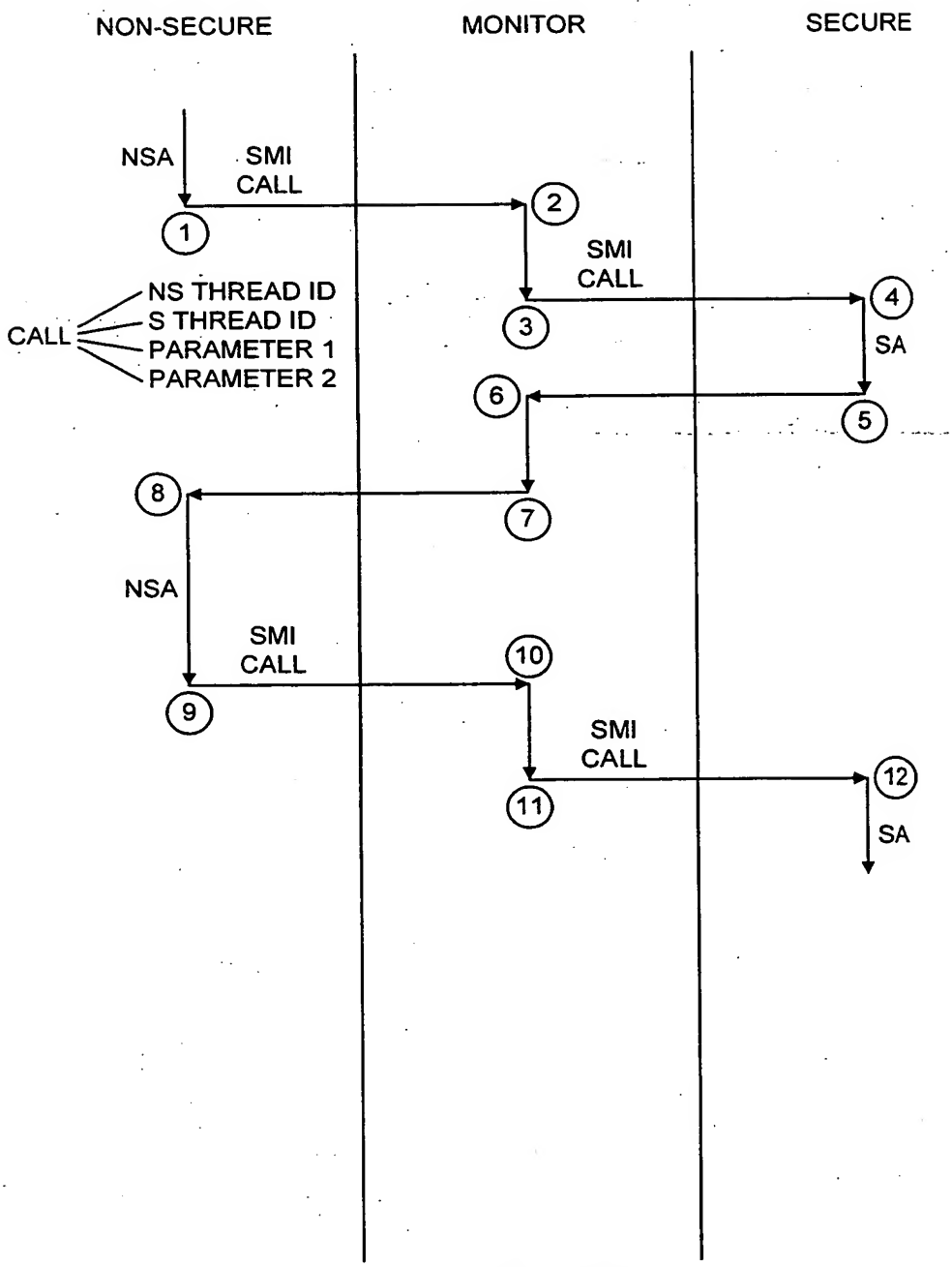
NON-SECURE
MONITOR
SECURE
NSA
SMI
CALL
1
2
SMI
CALL
3
4
SA
5
6
CALL
NS THREAD ID
S THREAD ID
PARAMETER 1
PARAMETER 2
7
8
NSA
SMI
CALL
9
10
SMI
CALL
11
12
SA

FIG. 29

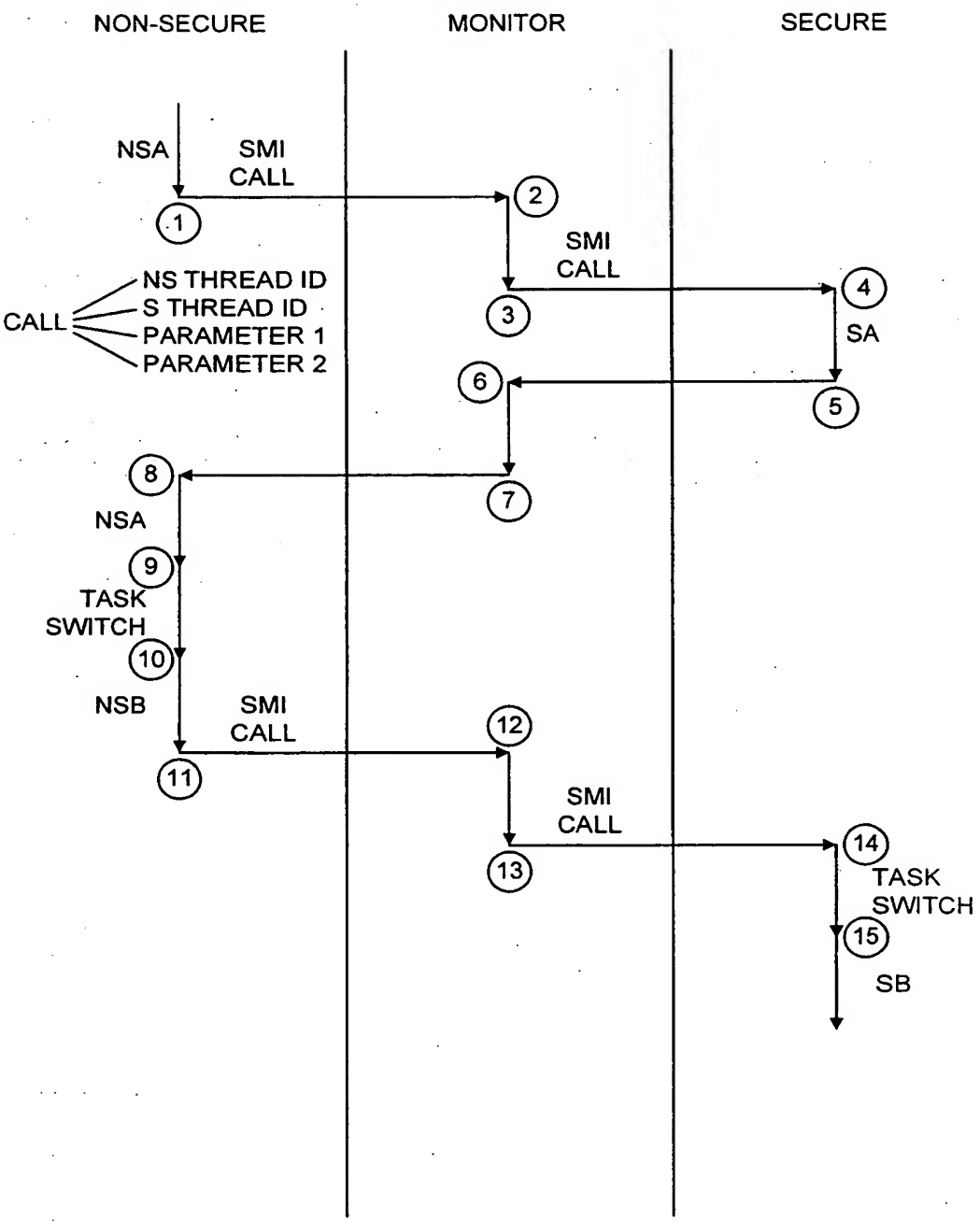
NON-SECURE
MONITOR
SECURE
NSA
SMI
CALL
1
2
SMI
CALL
3
4
SA
5
6
7
8
NSA
9
TASK
SWITCH
10
NSB
SMI
CALL
11
12
SMI
CALL
13
14
TASK
SWITCH
15
SB
CALL
NS THREAD ID
S THREAD ID
PARAMETER 1
PARAMETER 2

FIG. 30

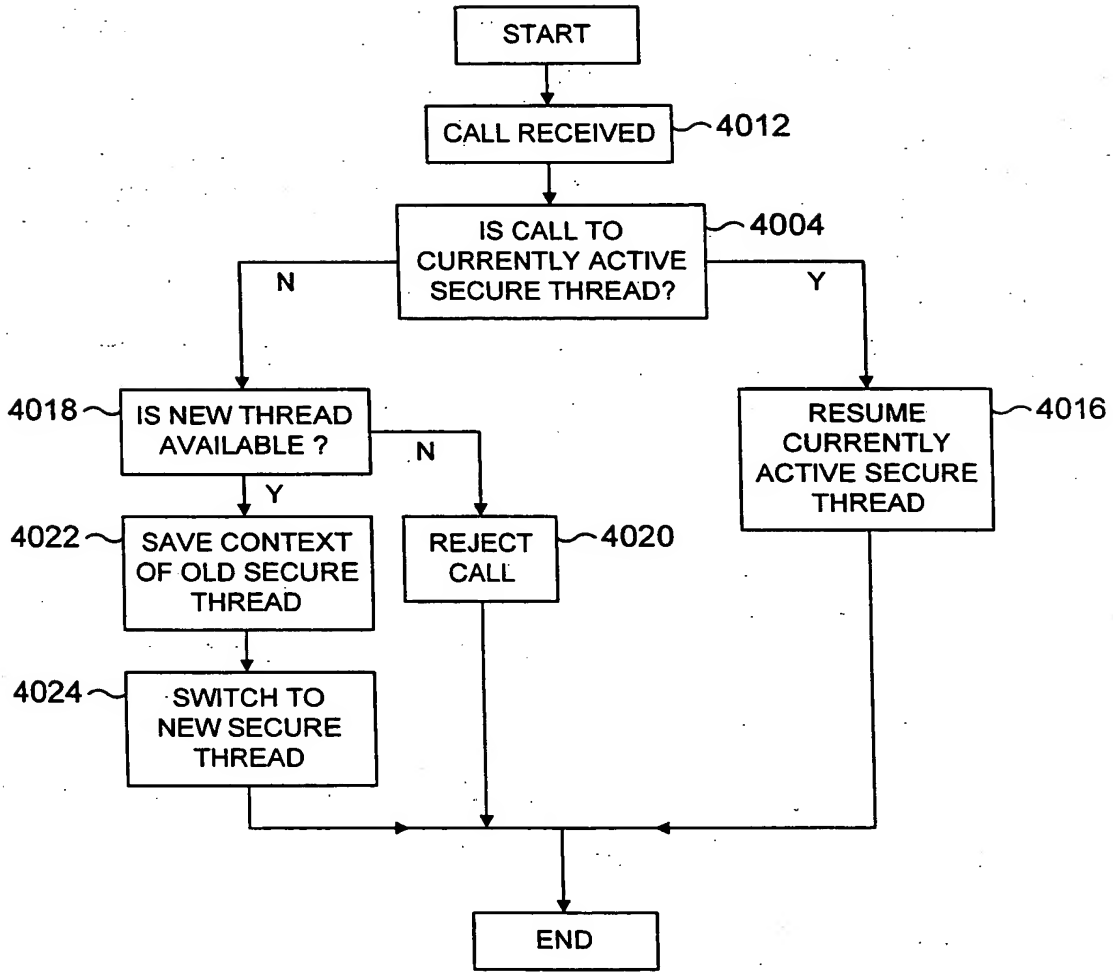
START
CALL RECEIVED
4012
IS CALL TO CURRENTLY ACTIVE SECURE THREAD?
4004
N
Y
4018
IS NEW THREAD AVAILABLE ?
N
Y
RESUME CURRENTLY ACTIVE SECURE THREAD
4016
4022
SAVE CONTEXT OF OLD SECURE THREAD
REJECT CALL
4020
4024
SWITCH TO NEW SECURE THREAD
END

FIG. 31

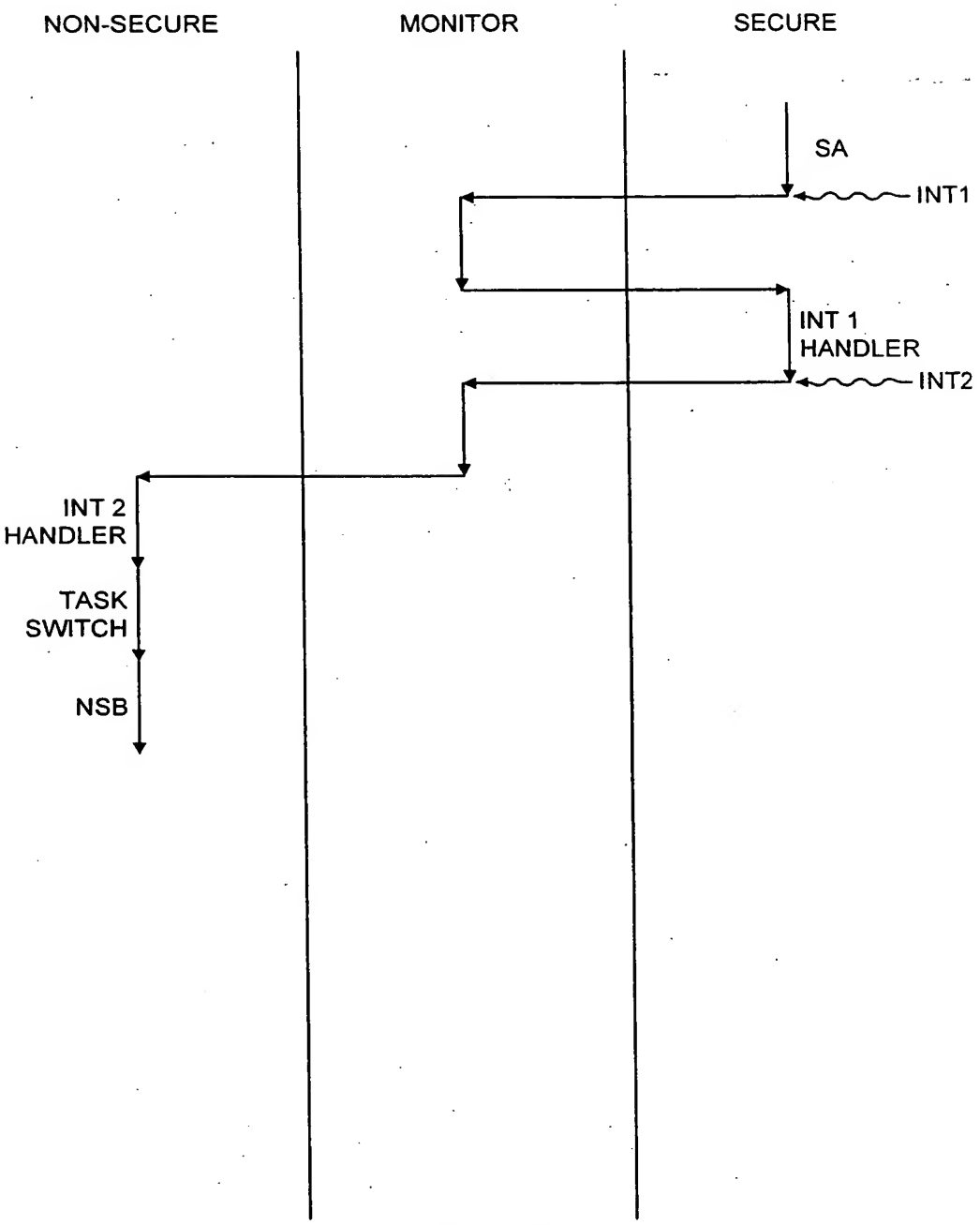
NON-SECURE
MONITOR
SECURE
SA
INT1
INT 1
HANDLER
INT2
INT 2
HANDLER
TASK
SWITCH
NSB

FIG. 32

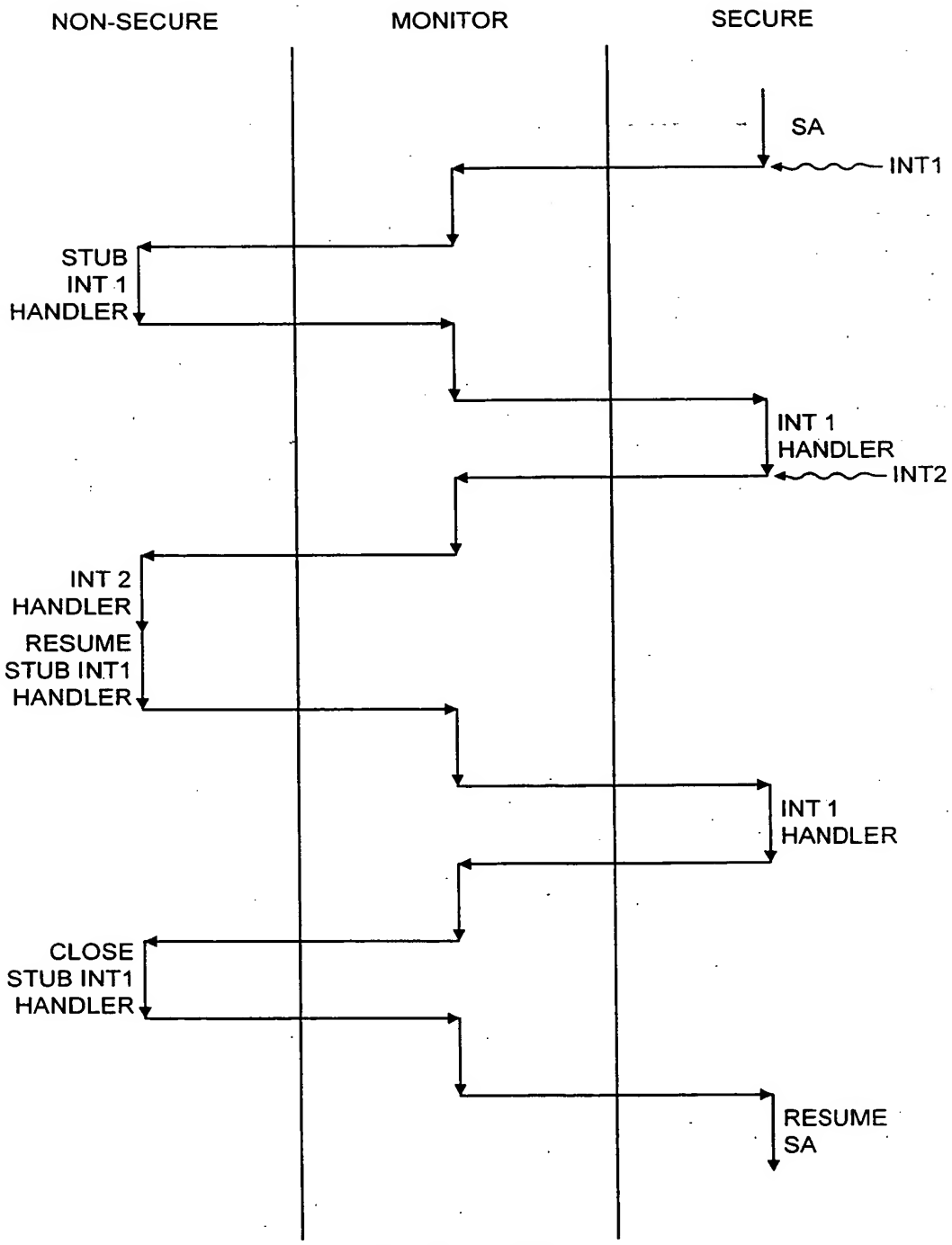
NON-SECURE
MONITOR
SECURE
SA
INT1
STUB
INT 1
HANDLER
INT 1
HANDLER
INT2
INT 2
HANDLER
RESUME
STUB INT1
HANDLER
INT 1
HANDLER
CLOSE
STUB INT1
HANDLER
RESUME
SA

FIG. 33

INTERRUPT TYPE/PRIORITY	HOW HANDLED
1	S
2	S
3	NS
4	NS/S
5	NS
6	NS/S
7	NS
•	•
•	•
•	•

NO S ONLY HANDLERS LOWER THAN HIGHEST NS HANDLER

FIG. 34

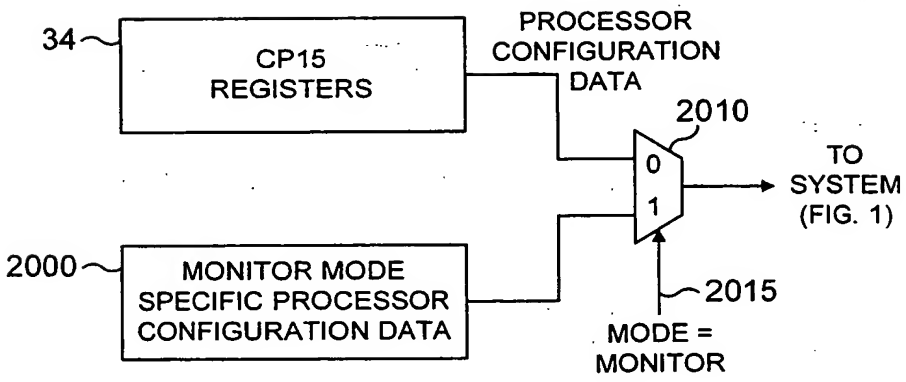

FIG. 35

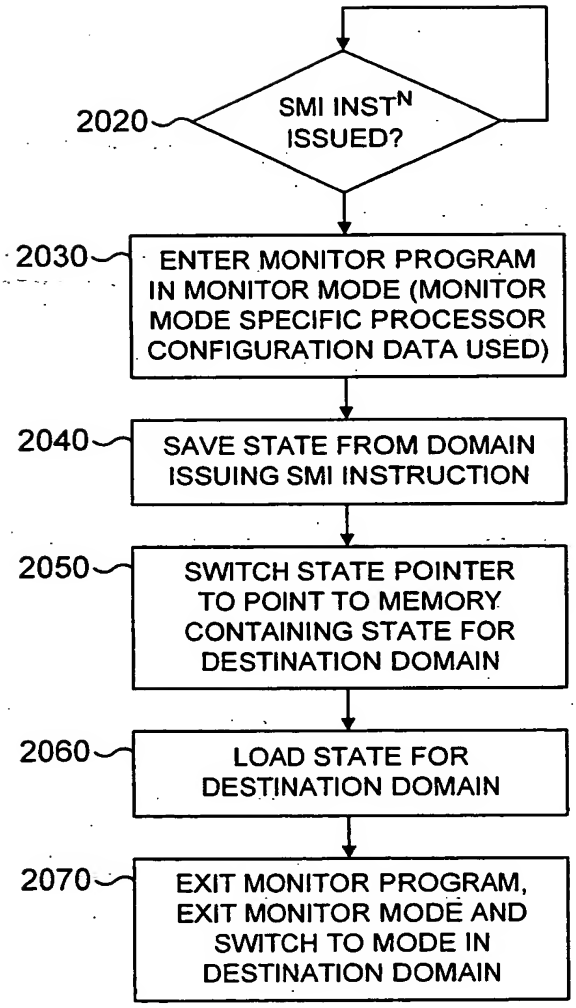
2020
SMI INST^N ISSUED?
2030
ENTER MONITOR PROGRAM IN MONITOR MODE (MONITOR MODE SPECIFIC PROCESSOR CONFIGURATION DATA USED)
2040
SAVE STATE FROM DOMAIN ISSUING SMI INSTRUCTION
2050
SWITCH STATE POINTER TO POINT TO MEMORY CONTAINING STATE FOR DESTINATION DOMAIN
2060
LOAD STATE FOR DESTINATION DOMAIN
2070
EXIT MONITOR PROGRAM, EXIT MONITOR MODE AND SWITCH TO MODE IN DESTINATION DOMAIN

FIG. 36

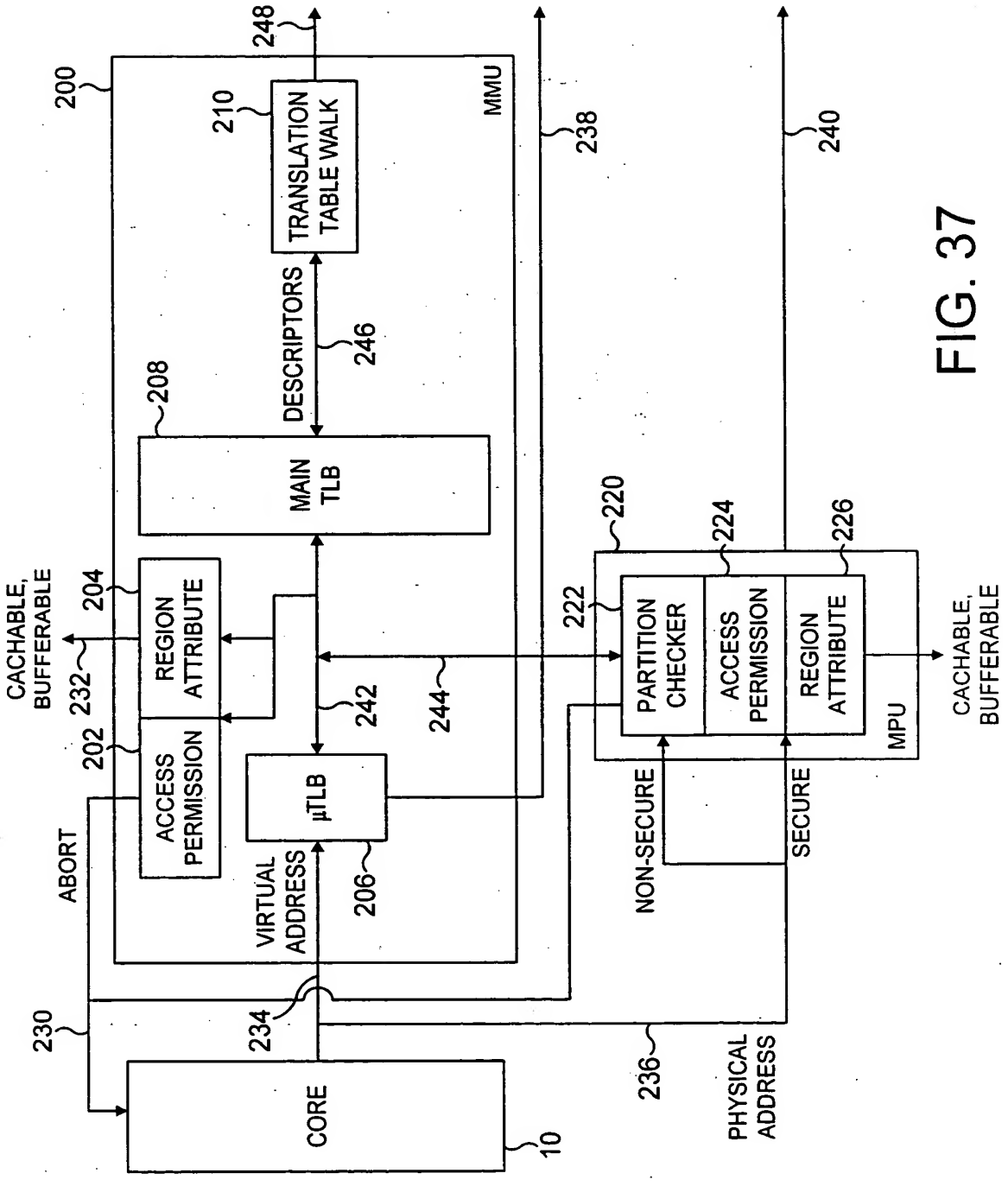
200
MMU
248
210
TRANSLATION TABLE WALK
238
240
DESCRIPTORS
246
208
MAIN TLB
220
224
226
222
PARTITION CHECKER
ACCESS PERMISSION
REGION ATTRIBUTE
MPU
CACHABLE, BUFFERABLE
204
232
REGION ATTRIBUTE
CACHABLE, BUFFERABLE
242
244
202
ACCESS PERMISSION
µTLB
206
ABORT
VIRTUAL ADDRESS
NON-SECURE
SECURE
230
234
236
PHYSICAL ADDRESS
CORE
10

FIG. 37

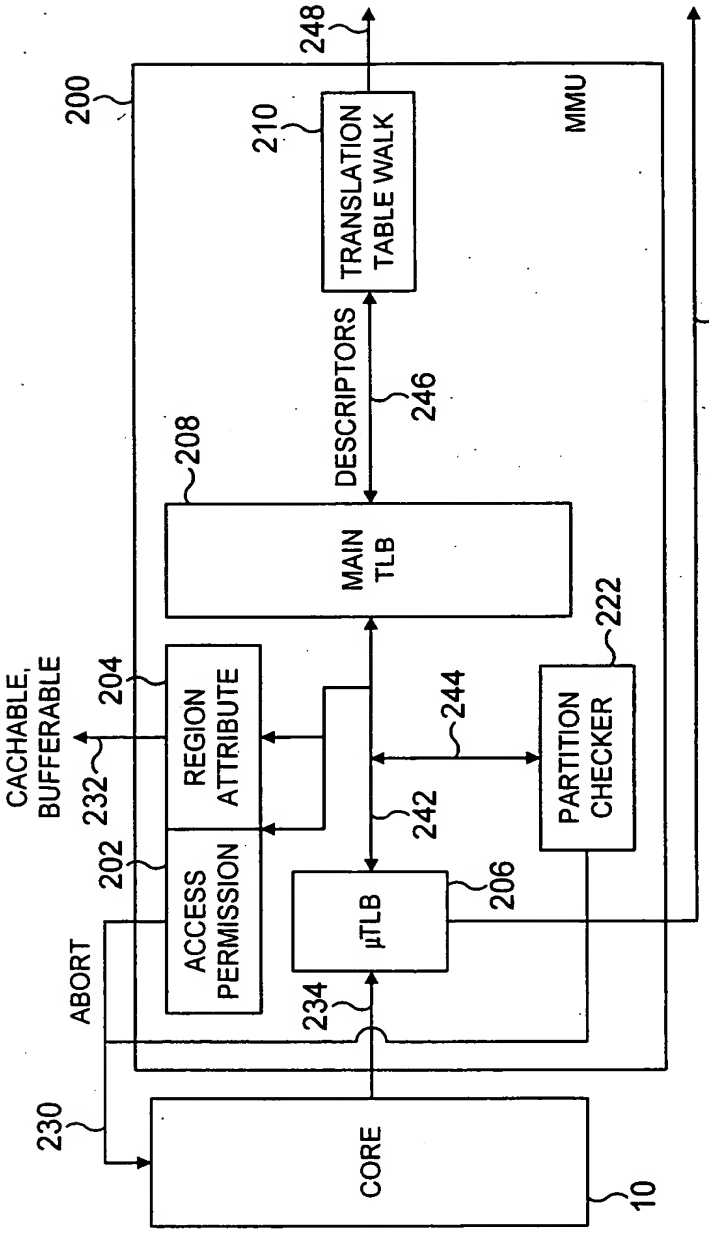
200
MMU
248
210
TRANSLATION TABLE WALK
DESCRIPTORS
246
208
MAIN TLB
238
222
PARTITION CHECKER
244
242
CACHABLE, BUFFERABLE
204
REGION ATTRIBUTE
232
202
ACCESS PERMISSION
206
µTLB
ABORT
234
230
CORE
10

FIG. 38

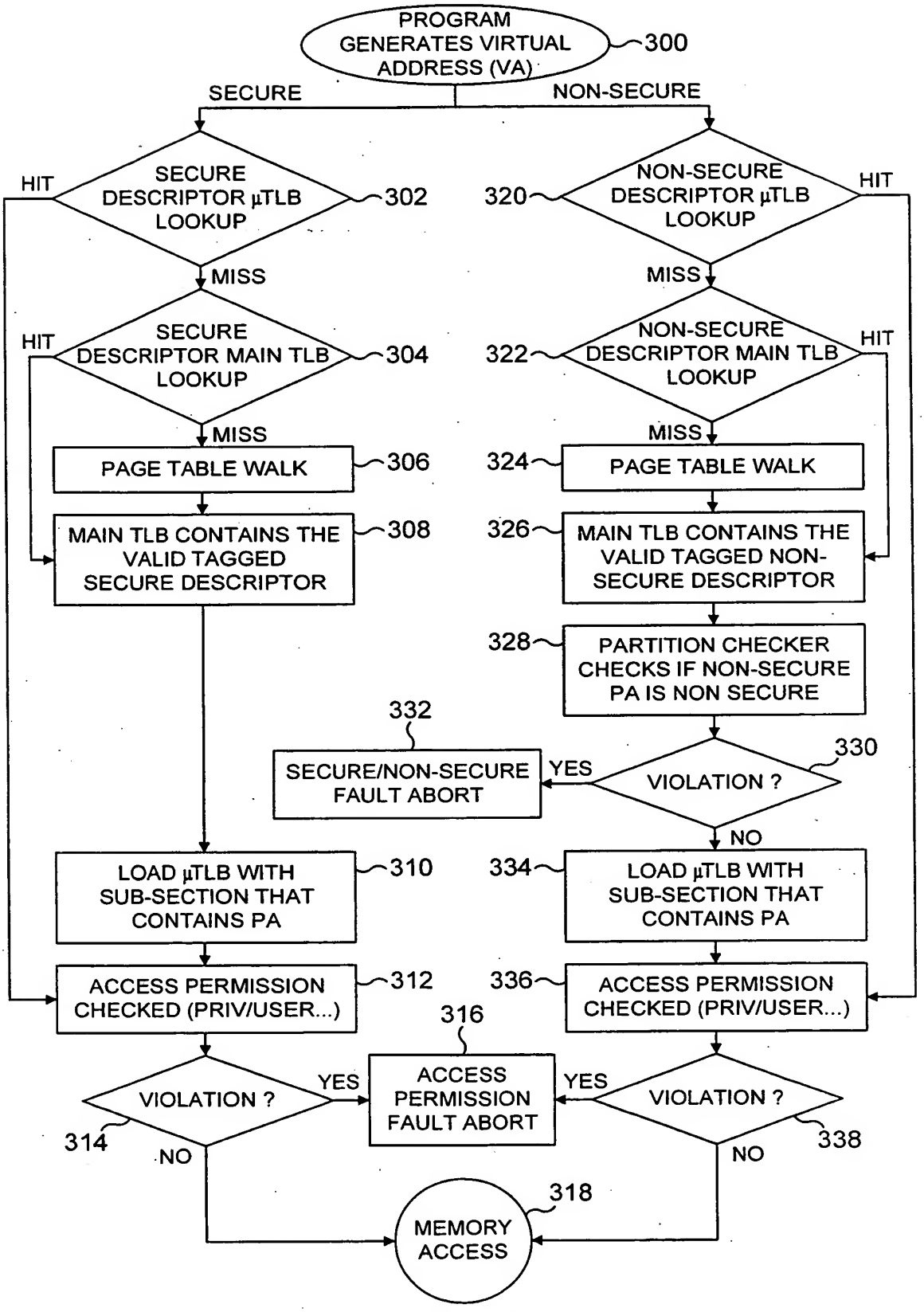
PROGRAM GENERATES VIRTUAL ADDRESS (VA)
300
SECURE
NON-SECURE
SECURE DESCRIPTOR µTLB LOOKUP
302
HIT
MISS
SECURE DESCRIPTOR MAIN TLB LOOKUP
304
HIT
MISS
PAGE TABLE WALK
306
MAIN TLB CONTAINS THE VALID TAGGED SECURE DESCRIPTOR
308
LOAD µTLB WITH SUB-SECTION THAT CONTAINS PA
310
ACCESS PERMISSION CHECKED (PRIV/USER...)
312
VIOLATION ?
314
YES
NO
NON-SECURE DESCRIPTOR µTLB LOOKUP
320
HIT
MISS
NON-SECURE DESCRIPTOR MAIN TLB LOOKUP
322
HIT
MISS
PAGE TABLE WALK
324
MAIN TLB CONTAINS THE VALID TAGGED NON-SECURE DESCRIPTOR
326
PARTITION CHECKER CHECKS IF NON-SECURE PA IS NON SECURE
328
VIOLATION ?
330
YES
NO
SECURE/NON-SECURE FAULT ABORT
332
LOAD µTLB WITH SUB-SECTION THAT CONTAINS PA
334
ACCESS PERMISSION CHECKED (PRIV/USER...)
336
VIOLATION ?
338
YES
NO
ACCESS PERMISSION FAULT ABORT
316
MEMORY ACCESS
318

FIG. 39

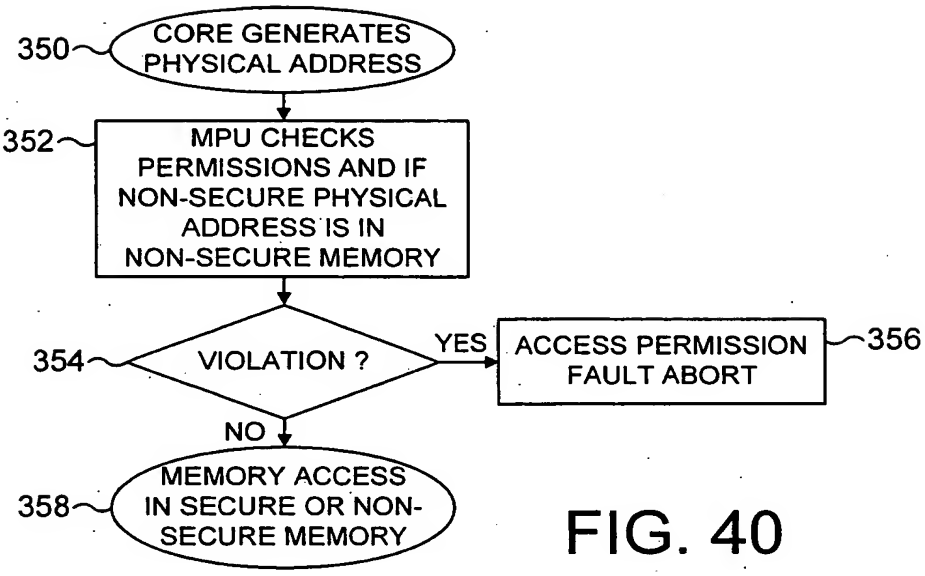
350
CORE GENERATES PHYSICAL ADDRESS
352
MPU CHECKS PERMISSIONS AND IF NON-SECURE PHYSICAL ADDRESS IS IN NON-SECURE MEMORY
354
VIOLATION ?
YES
ACCESS PERMISSION FAULT ABORT
356
NO
358
MEMORY ACCESS IN SECURE OR NON-SECURE MEMORY

FIG. 40

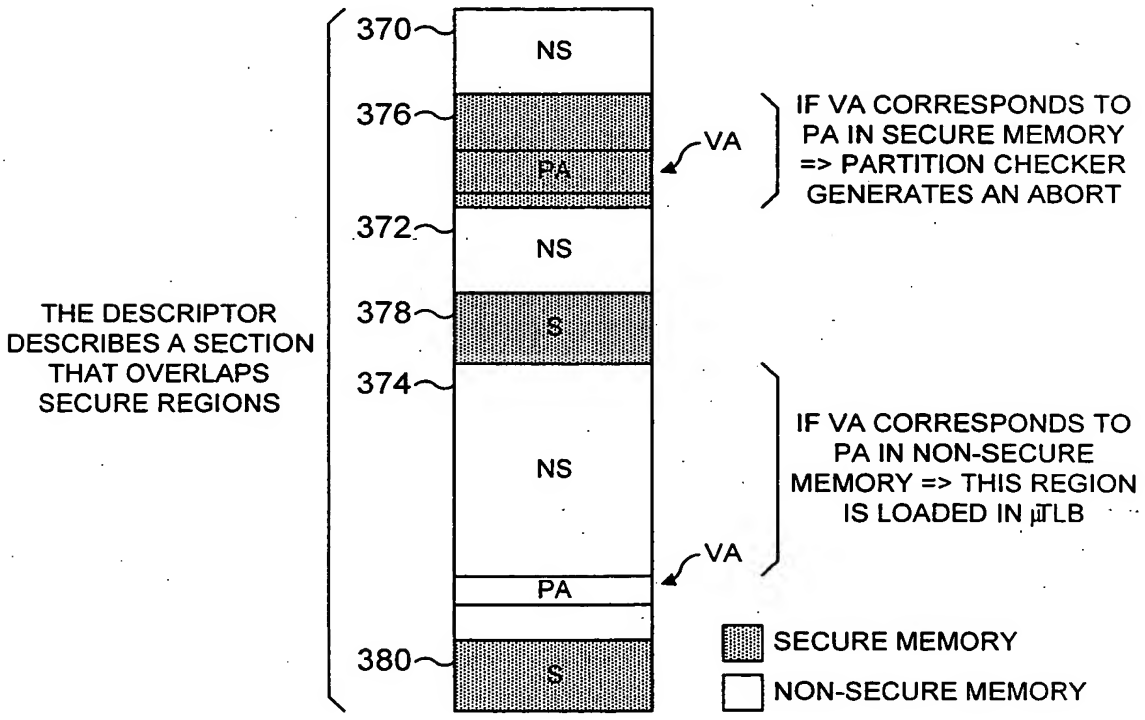
370
NS
376
PA
VA
IF VA CORRESPONDS TO PA IN SECURE MEMORY => PARTITION CHECKER GENERATES AN ABORT
372
NS
THE DESCRIPTOR DESCRIBES A SECTION THAT OVERLAPS SECURE REGIONS
378
S
374
NS
IF VA CORRESPONDS TO PA IN NON-SECURE MEMORY => THIS REGION IS LOADED IN μTLB
VA
PA
380
S
SECURE MEMORY
NON-SECURE MEMORY

FIG. 41

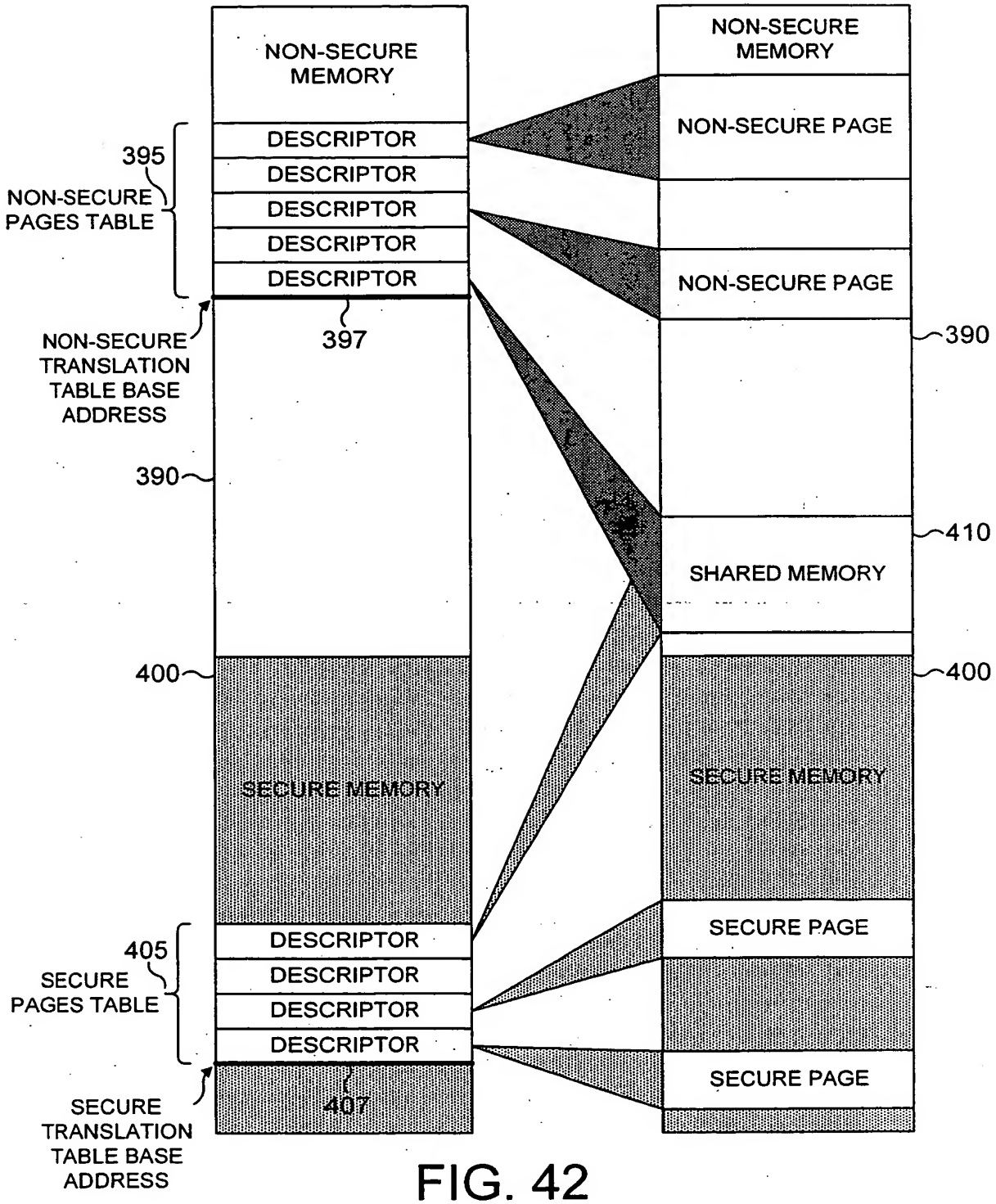
NON-SECURE MEMORY
DESCRIPTOR
DESCRIPTOR
DESCRIPTOR
DESCRIPTOR
DESCRIPTOR
395
NON-SECURE PAGES TABLE
397
NON-SECURE TRANSLATION TABLE BASE ADDRESS
390
400
SECURE MEMORY
DESCRIPTOR
DESCRIPTOR
DESCRIPTOR
DESCRIPTOR
405
SECURE PAGES TABLE
407
SECURE TRANSLATION TABLE BASE ADDRESS
NON-SECURE MEMORY
NON-SECURE PAGE
NON-SECURE PAGE
390
410
SHARED MEMORY
400
SECURE MEMORY
SECURE PAGE
SECURE PAGE

FIG. 42

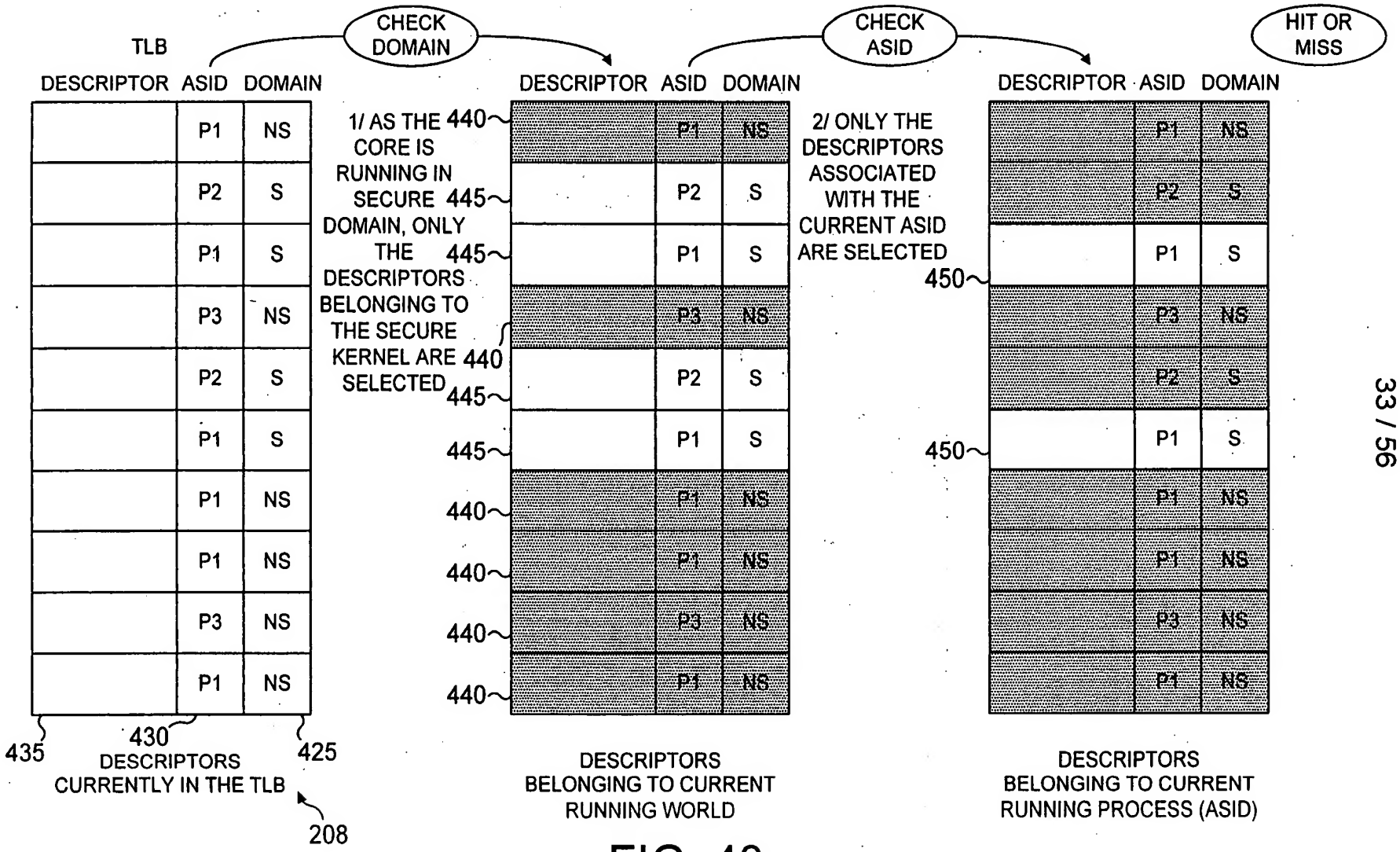

FIG. 43

Inventor: MANSELL et al
SN 10/713,454/Sheet 33 of 56
Atty. Dkt.: 550-485

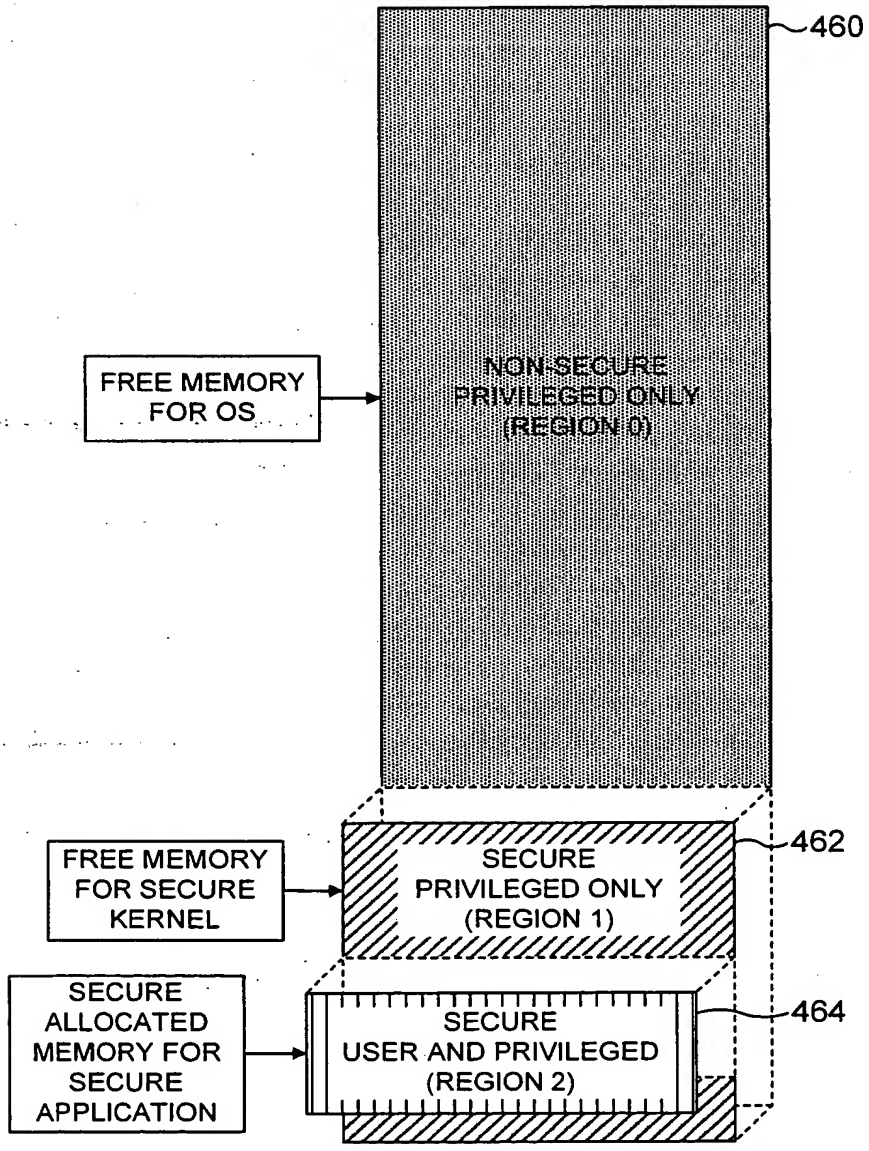
460
FREE MEMORY FOR OS
NON-SECURE PRIVILEGED ONLY (REGION 0)
FREE MEMORY FOR SECURE KERNEL
SECURE PRIVILEGED ONLY (REGION 1)
462
SECURE ALLOCATED MEMORY FOR SECURE APPLICATION
SECURE USER AND PRIVILEGED (REGION 2)
464

FIG. 44

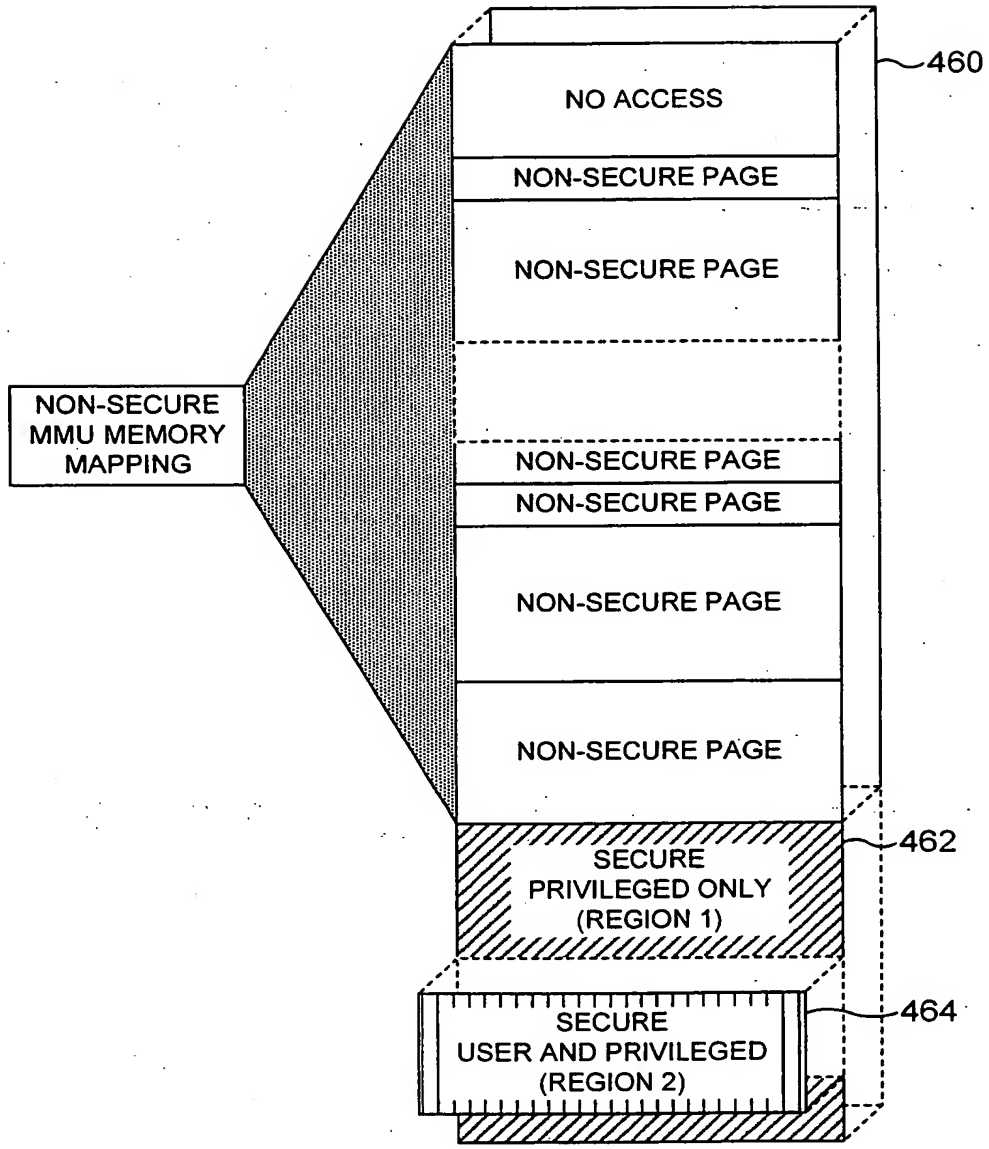
460
NO ACCESS
NON-SECURE PAGE
NON-SECURE PAGE
NON-SECURE MMU MEMORY MAPPING
NON-SECURE PAGE
NON-SECURE PAGE
NON-SECURE PAGE
NON-SECURE PAGE
462
SECURE PRIVILEGED ONLY (REGION 1)
464
SECURE USER AND PRIVILEGED (REGION 2)

FIG. 45

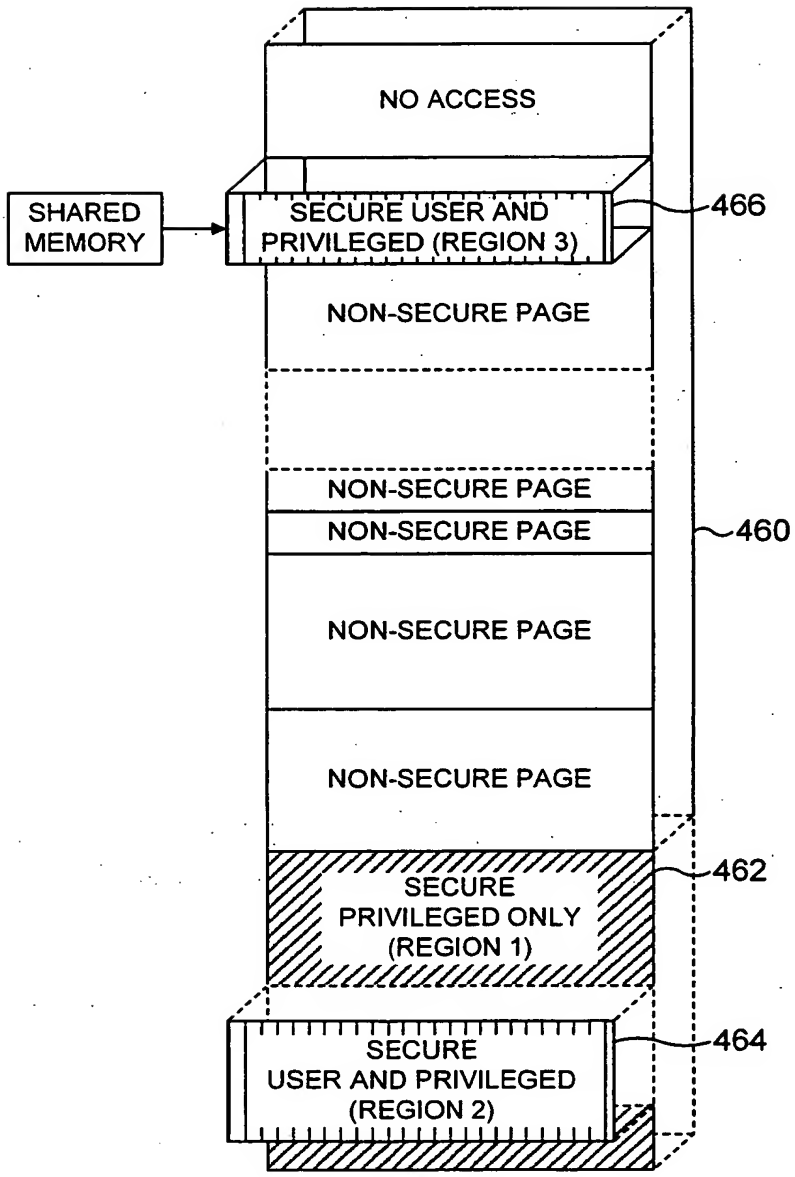
NO ACCESS
SHARED MEMORY
SECURE USER AND PRIVILEGED (REGION 3)
466
NON-SECURE PAGE
NON-SECURE PAGE
NON-SECURE PAGE
460
NON-SECURE PAGE
NON-SECURE PAGE
SECURE PRIVILEGED ONLY (REGION 1)
462
SECURE USER AND PRIVILEGED (REGION 2)
464

FIG. 46

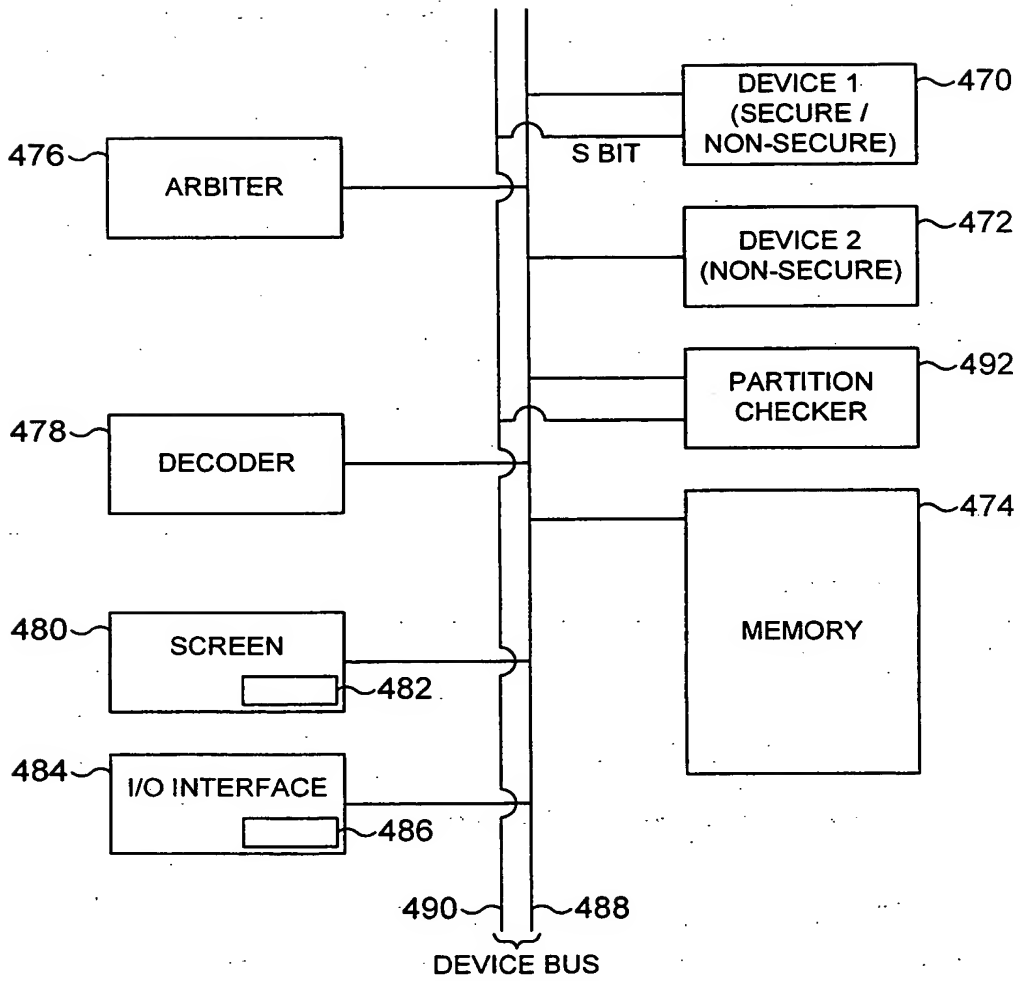
DEVICE 1
(SECURE /
NON-SECURE)
470
S BIT
476
ARBITER
DEVICE 2
(NON-SECURE)
472
PARTITION
CHECKER
492
478
DECODER
474
MEMORY
480
SCREEN
482
484
I/O INTERFACE
486
490
488
DEVICE BUS

FIG. 47

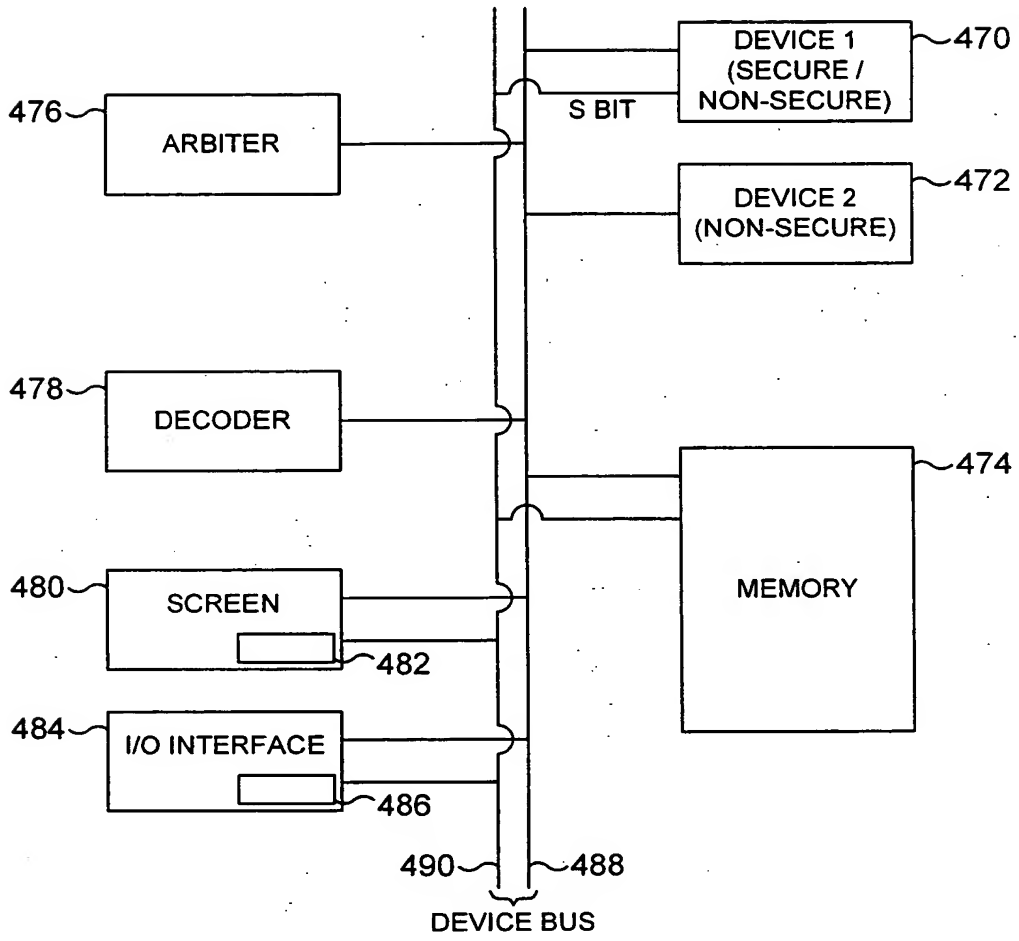
DEVICE 1
(SECURE /
NON-SECURE)
470
S BIT
476
ARBITER
DEVICE 2
(NON-SECURE)
472
478
DECODER
474
MEMORY
480
SCREEN
482
484
I/O INTERFACE
486
490
488
DEVICE BUS

FIG. 48

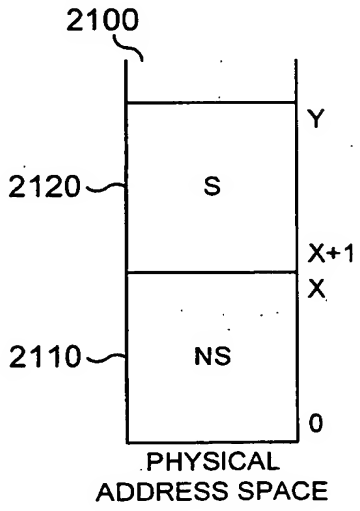
2100
Y
2120
S
X+1
X
2110
NS
0
PHYSICAL
ADDRESS SPACE

FIG. 49

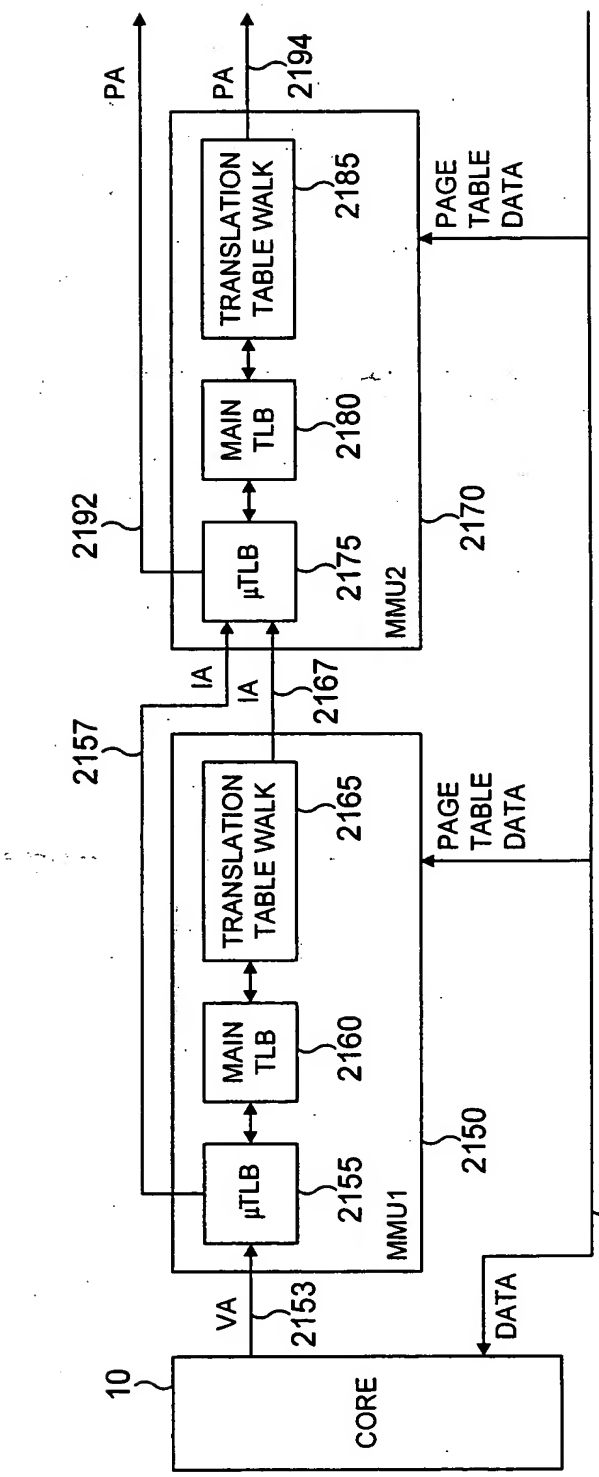
PA
PA
2194
TRANSLATION TABLE WALK
2185
PAGE TABLE DATA
MAIN TLB
2180
2192
µTLB
2175
MMU2
2170
IA
IA
2167
2157
TRANSLATION TABLE WALK
2165
PAGE TABLE DATA
MAIN TLB
2160
µTLB
2155
MMU1
2150
2190
VA
2153
DATA
10
CORE

FIG. 50A

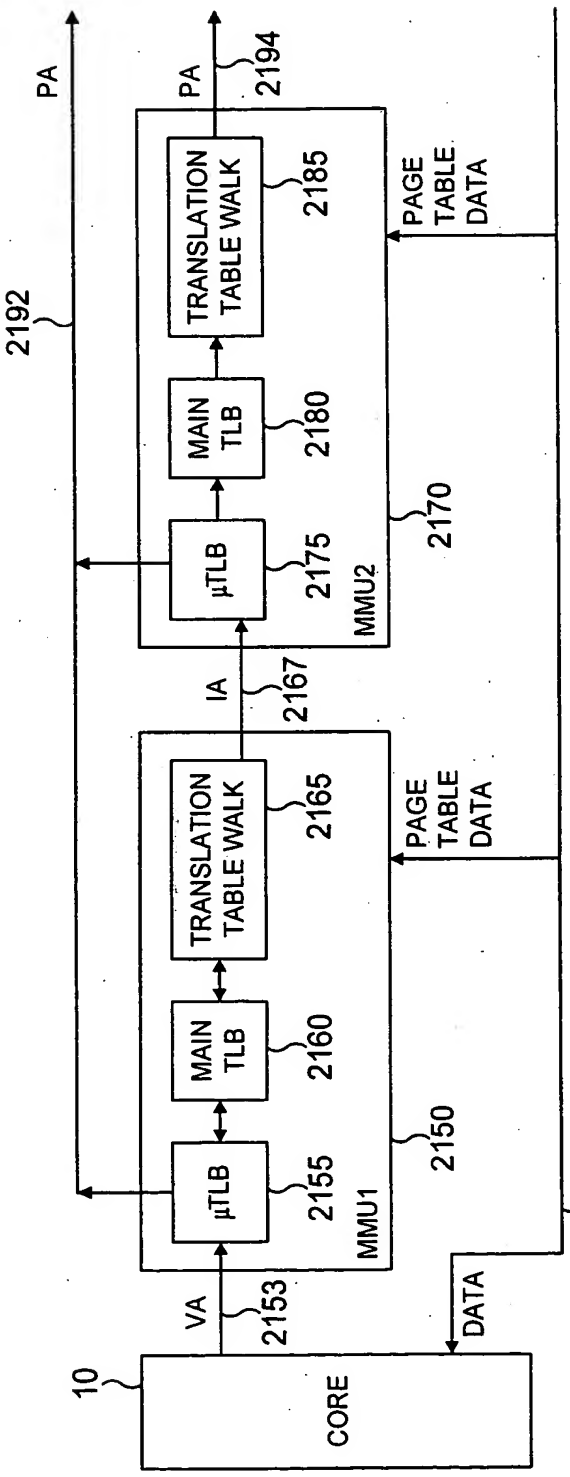
10
CORE
VA
2153
µTLB
2155
MAIN TLB
2160
TRANSLATION TABLE WALK
2165
MMU1
2150
IA
2167
µTLB
2175
MAIN TLB
2180
TRANSLATION TABLE WALK
2185
MMU2
2170
PA
2192
PA
2194
PAGE TABLE DATA
PAGE TABLE DATA
DATA
2190

FIG. 50B

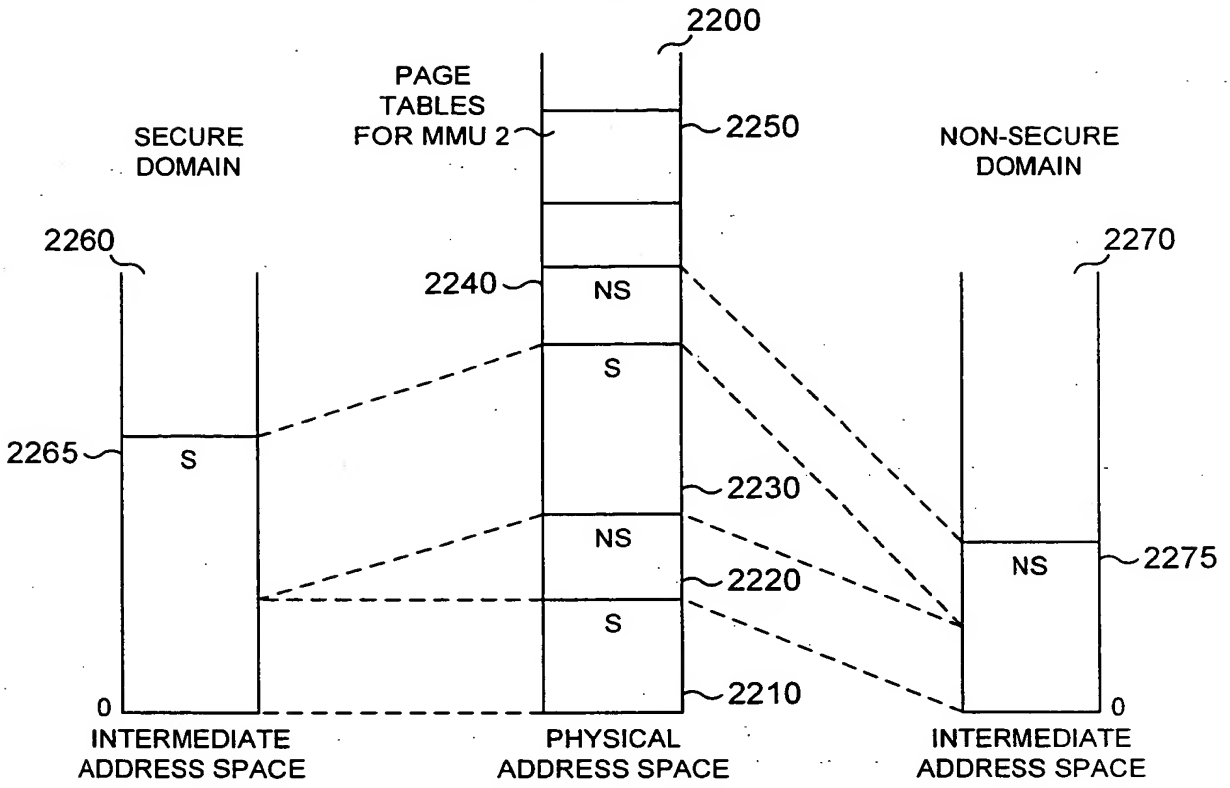
2200
PAGE TABLES FOR MMU 2
2250
SECURE DOMAIN
NON-SECURE DOMAIN
2260
2270
2240
NS
S
2265
S
2230
NS
2220
NS
2275
S
2210
0
0
INTERMEDIATE ADDRESS SPACE
PHYSICAL ADDRESS SPACE
INTERMEDIATE ADDRESS SPACE

FIG. 51

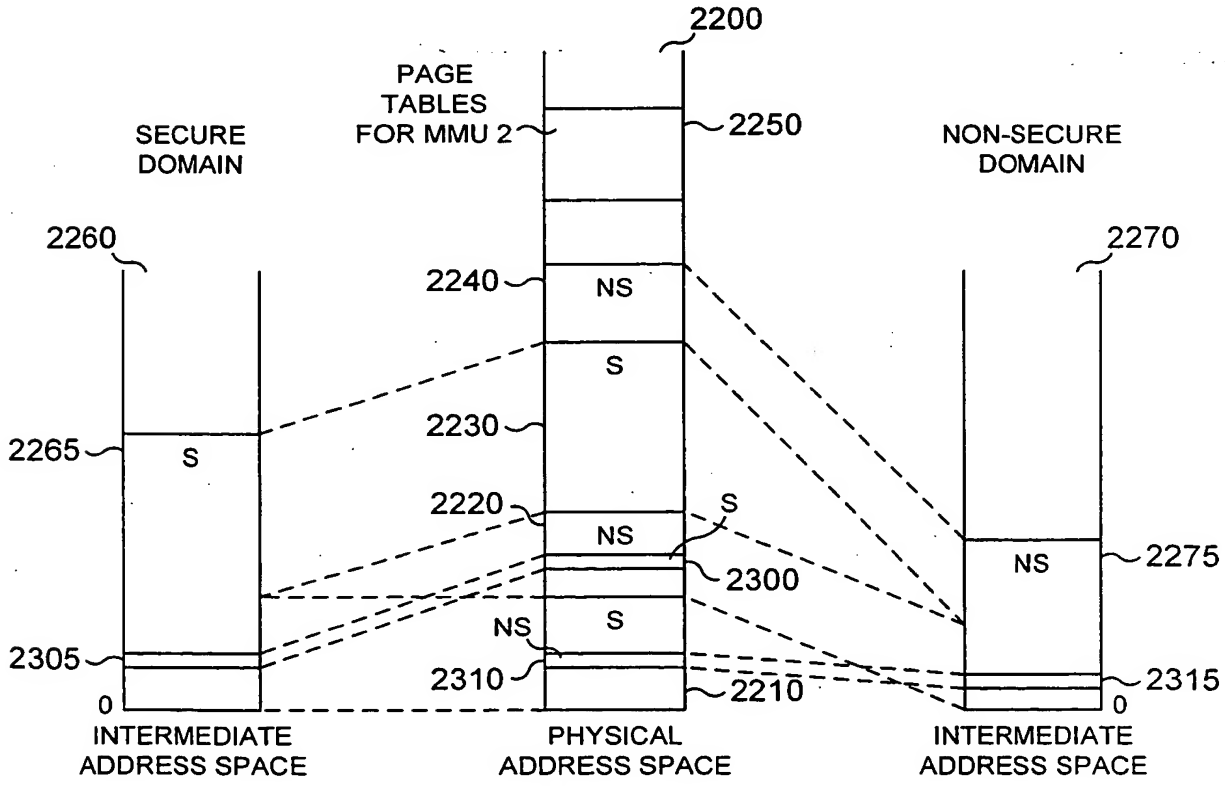
2200
PAGE TABLES FOR MMU 2
2250
SECURE DOMAIN
NON-SECURE DOMAIN
2260
2270
2240
NS
S
2230
2265
S
2220
S
NS
NS
2275
2300
NS
S
2305
2310
2210
2315
0
0
INTERMEDIATE ADDRESS SPACE
PHYSICAL ADDRESS SPACE
INTERMEDIATE ADDRESS SPACE

FIG. 52

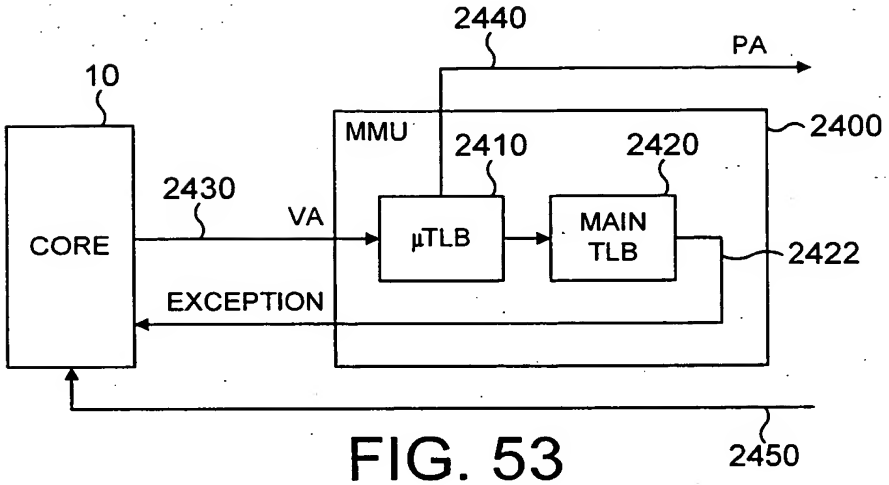
2440
PA
10
MMU
2410
2420
2400
2430
VA
CORE
µTLB
MAIN
TLB
2422
EXCEPTION
2450

FIG. 53

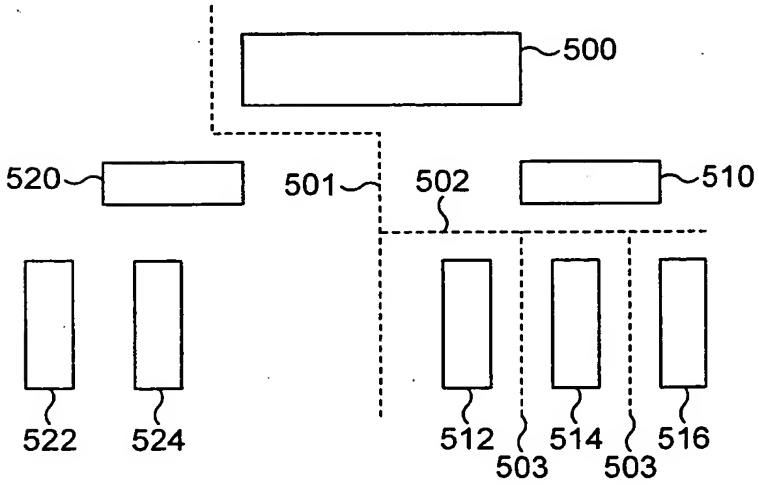
500
520
501
502
510
522
524
512
503
514
503
516

FIG. 59

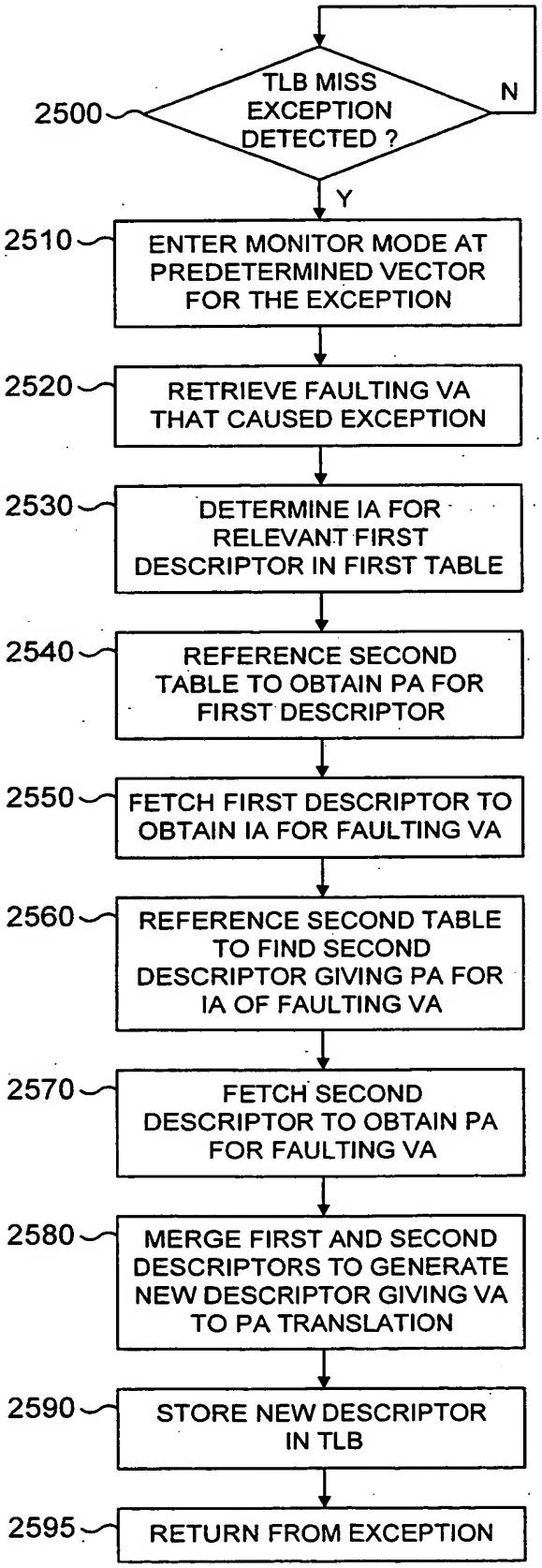
2500
TLB MISS EXCEPTION DETECTED ?
N
Y
2510
ENTER MONITOR MODE AT PREDETERMINED VECTOR FOR THE EXCEPTION
2520
RETRIEVE FAULTING VA THAT CAUSED EXCEPTION
2530
DETERMINE IA FOR RELEVANT FIRST DESCRIPTOR IN FIRST TABLE
2540
REFERENCE SECOND TABLE TO OBTAIN PA FOR FIRST DESCRIPTOR
2550
FETCH FIRST DESCRIPTOR TO OBTAIN IA FOR FAULTING VA
2560
REFERENCE SECOND TABLE TO FIND SECOND DESCRIPTOR GIVING PA FOR IA OF FAULTING VA
2570
FETCH SECOND DESCRIPTOR TO OBTAIN PA FOR FAULTING VA
2580
MERGE FIRST AND SECOND DESCRIPTORS TO GENERATE NEW DESCRIPTOR GIVING VA TO PA TRANSLATION
2590
STORE NEW DESCRIPTOR IN TLB
2595
RETURN FROM EXCEPTION

FIG. 54

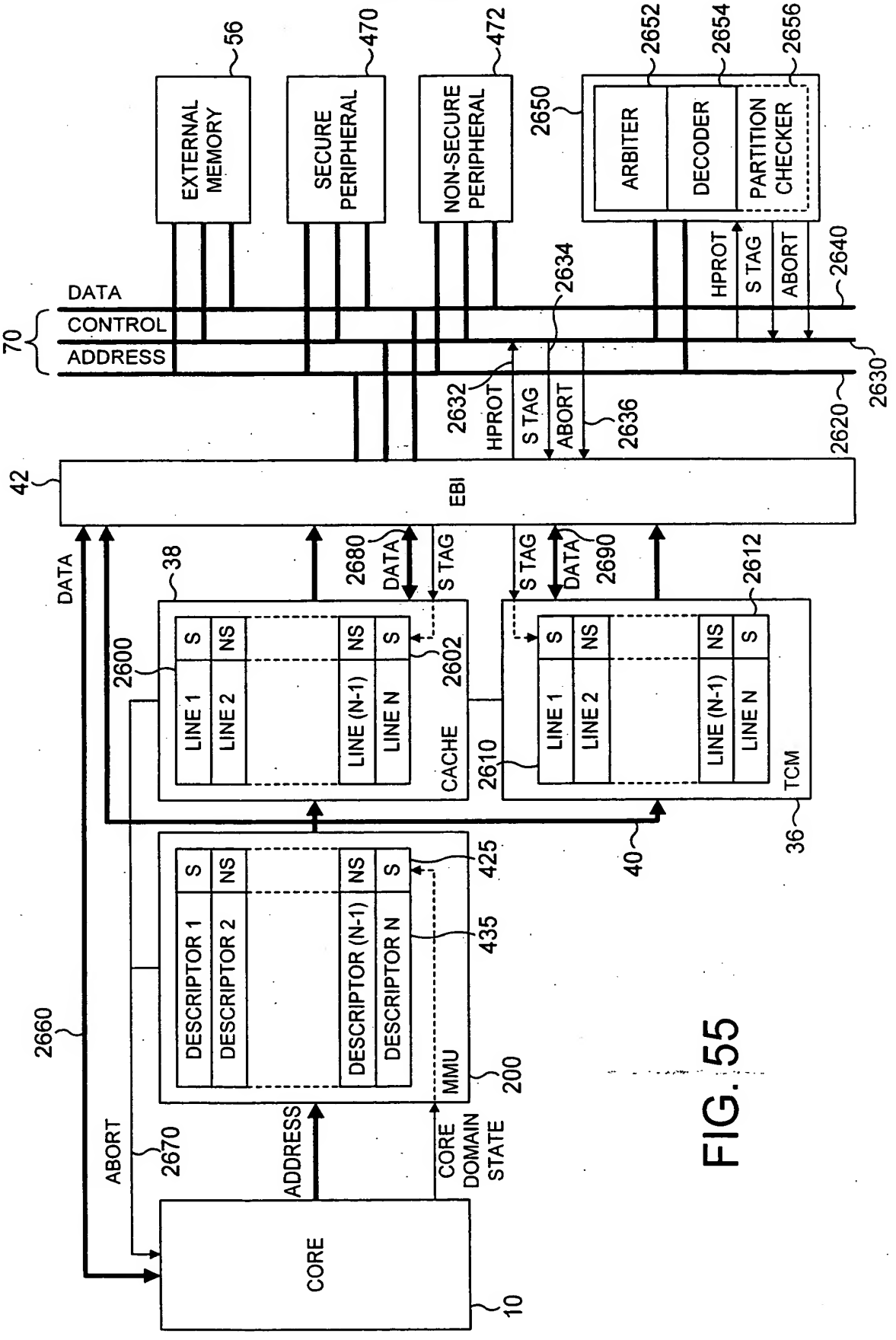
EXTERNAL MEMORY
56
SECURE PERIPHERAL
470
NON-SECURE PERIPHERAL
472
ARBITER
DECODER
PARTITION CHECKER
2650
2652
2654
2656
DATA
CONTROL
ADDRESS
70
2634
HPROT
S TAG
ABORT
2640
2632
2636
2630
2620
42
EBI
2680
DATA
S TAG
2690
38
2600
2602
LINE 1
LINE 2
LINE (N-1)
LINE N
S
NS
CACHE
2610
2612
TCM
36
40
DESCRIPTOR 1
DESCRIPTOR 2
DESCRIPTOR (N-1)
DESCRIPTOR N
425
435
MMU
200
2660
2670
ABORT
ADDRESS
CORE DOMAIN STATE
CORE
10

FIG. 55

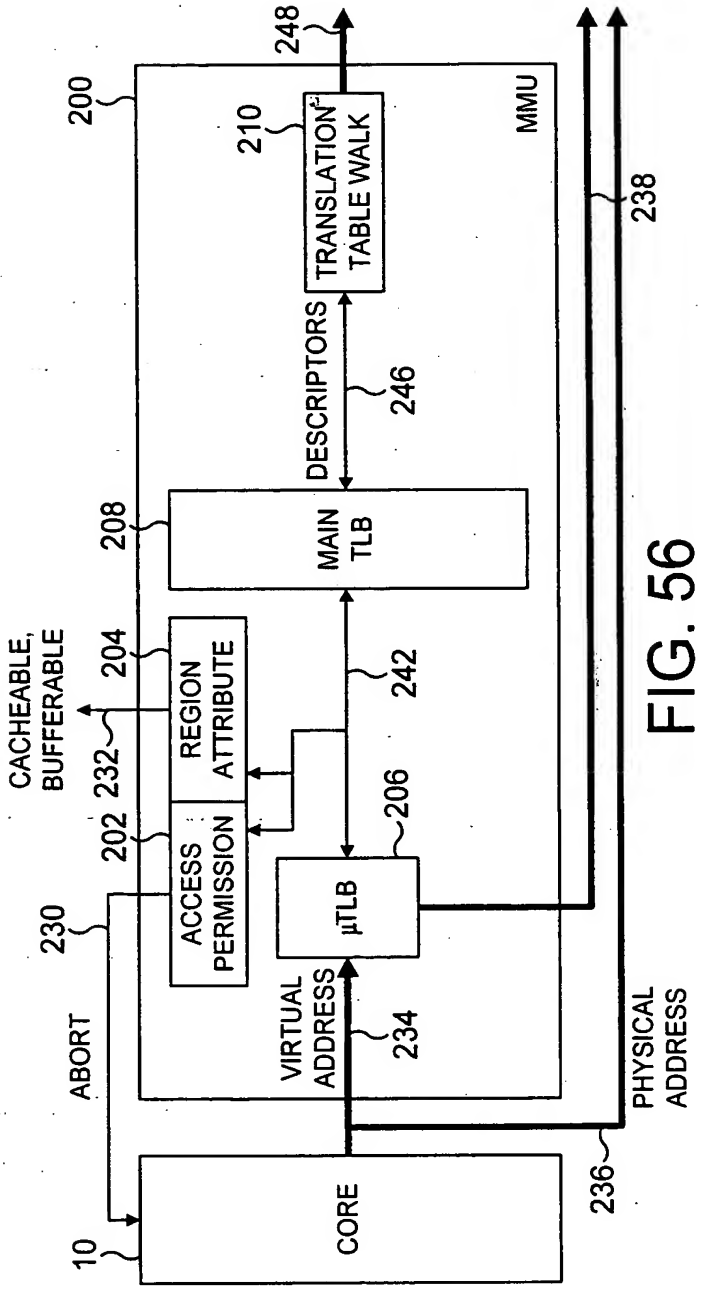
CORE
10
ABORT
230
ACCESS PERMISSION
202
REGION ATTRIBUTE
232
CACHEABLE, BUFFERABLE
204
VIRTUAL ADDRESS
234
µTLB
206
MAIN TLB
208
DESCRIPTORS
246
TRANSLATION TABLE WALK
210
MMU
200
248
242
238
PHYSICAL ADDRESS
236

FIG. 56

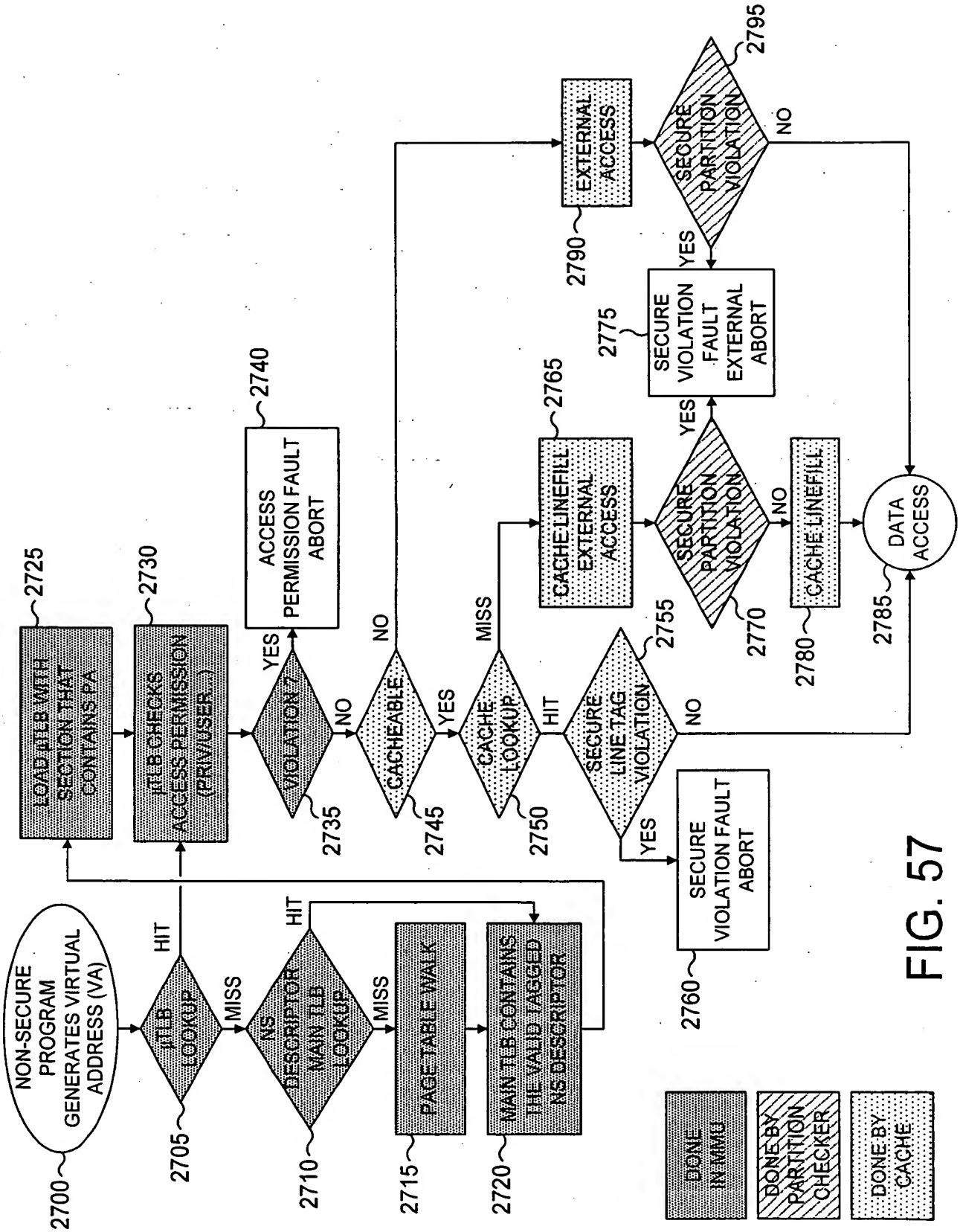
NON-SECURE PROGRAM GENERATES VIRTUAL ADDRESS (VA)
2700
μTLB LOOKUP
2705
HIT
MISS
NS DESCRIPTOR MAIN TLB LOOKUP
2710
HIT
MISS
PAGE TABLE WALK
2715
MAIN TLB CONTAINS THE VALID TAGGED NS DESCRIPTOR
2720
LOAD TLB WITH SECTION THAT CONTAINS PA
2725
μTLB CHECKS ACCESS PERMISSION (PRIV/USER...)
2730
VIOLATION ?
2735
YES
NO
ACCESS PERMISSION FAULT ABORT
2740
CACHEABLE
2745
YES
NO
CACHE LOOKUP
2750
MISS
HIT
SECURE LINE TAG VIOLATION
2755
YES
NO
SECURE VIOLATION FAULT ABORT
2760
CACHE LINEFILL EXTERNAL ACCESS
2765
SECURE PARTITION VIOLATION
2770
YES
NO
SECURE VIOLATION FAULT EXTERNAL ABORT
2775
CACHE LINEFILL
2780
EXTERNAL ACCESS
2790
SECURE PARTITION VIOLATION
2795
YES
NO
DATA ACCESS
2785
DONE IN MMU
DONE BY PARTITION CHECKER
DONE BY CACHE

FIG. 57

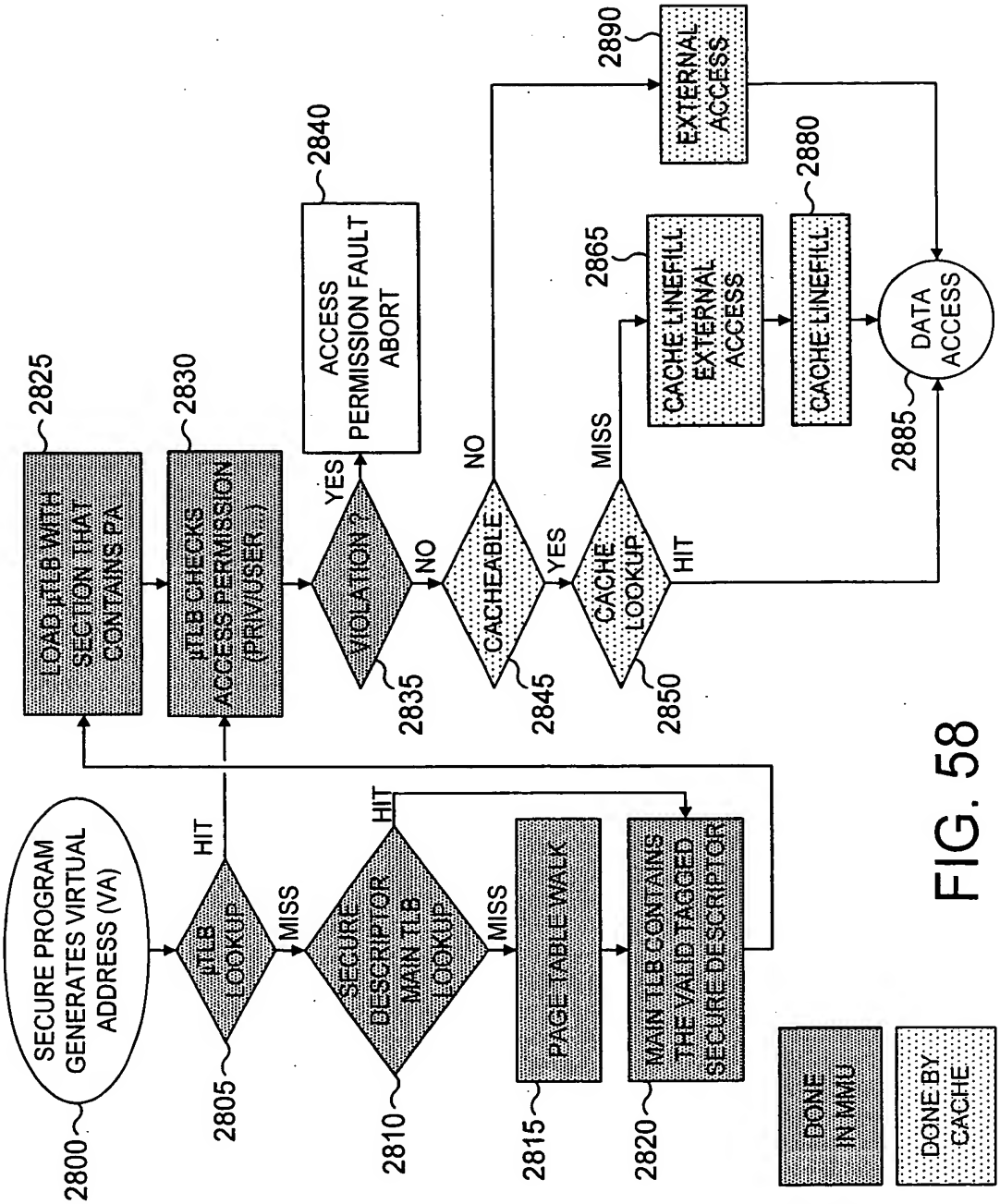
SECURE PROGRAM GENERATES VIRTUAL ADDRESS (VA)
2800
PTLB LOOKUP
HIT
MISS
2805
SECURE DESCRIPTOR MAIN TLB LOOKUP
HIT
MISS
2810
PAGE TABLE WALK
2815
MAIN TLB CONTAINS THE VALID TAGGED SECURE DESCRIPTOR
2820
DONE IN MMU
DONE BY CACHE
LOAD TLB WITH SECTION THAT CONTAINS PA
2825
PTLB CHECKS ACCESS PERMISSION (PRIV/USER...)
2830
VIOLATION ?
YES
NO
2835
ACCESS PERMISSION FAULT ABORT
2840
CACHEABLE
NO
YES
2845
CACHE LOOKUP
MISS
HIT
2850
EXTERNAL ACCESS
2890
CACHE LINEFILL EXTERNAL ACCESS
2865
CACHE LINEFILL
2880
DATA ACCESS
2885

FIG. 58

METHOD OF ENTRY	HOW TO PROGRAM?	HOW TO ENTER?	ENTRY MODE
BREAKPOINT HITS	DEBUG TAP OR SOFTWARE (CP14)	PROGRAM BREAKPOINT REGISTER AND/OR CONTEXT-ID REGISTER AND COMPARISONS SUCCEED WITH INSTRUCTION ADDRESS AND/OR CP15 CONTEXT ID ([2])	HALT/MONITOR ([1])
SOFTWARE BREAKPOINT INSTRUCTION	PUT A BKPT INSTRUCTION INTO SCAN CHAIN 4 (INSTRUCTION TRANSFER REGISTER) THROUGH DEBUG TAP OR USE BKPT INSTRUCTION DIRECTLY IN THE CODE	BKPT INSTRUCTION MUST REACH EXECUTION STAGE	HALT/MONITOR
VECTOR TRAP BREAKPOINT	DEBUG TAP	PROGRAM VECTOR TRAP REGISTER AND ADDRESS MATCHES	HALT/MONITOR
WATCHPOINT HITS	DEBUG TAP OR SOFTWARE (CP14)	PROGRAM WATCHPOINT REGISTER AND/OR CONTEXT-ID REGISTER AND COMPARISONS SUCCEED WITH INSTRUCTION ADDRESS AND/OR CP15 CONTEXT ID ([2])	HALT/MONITOR ([1])
INTERNAL DEBUG REQUEST	DEBUG TAP	HALT INSTRUCTION HAS BEEN SCANNED IN	HALT
EXTERNAL DEBUG REQUEST	NOT APPLICABLE	EDBGRQ INPUT PIN IS ASSERTED	HALT

([1]): IN MONITOR MODE, BREAKPOINTS AND WATCHPOINTS CANNOT BE DATA-DEPENDENT.

([2]): THE CORES HAVE SUPPORT FOR THREAD-AWARE BREAKPOINTS AND WATCHPOINTS IN ORDER TO ABLE TO ENABLE SECURE DEBUG ON SOME PARTICULAR THREADS.

FIG. 60

Inventor: MANSELL et al
SN 10/713,454/Sheet 48 of 56
Atty. Dkt.: 550-485

NAME	MEANING	RESET VALUE	ACCESS	INSERTED IN SCAN CHAIN FOR TEST
MONITOR MODE ENABLE BIT	0: HALT MODE 1: MONITOR MODE	1	R/W BY PROGRAMMING THE ICE BY THE JTAG (SCAN 1) •R/W BY USING MRC/MCR INSTRUCTION (CP14)	YES
SECURE DEBUG ENABLE BIT	0: DEBUG IN NON-SECURE WORLD ONLY 1: DEBUG IN SECURE WORLD AND NON-SECURE WORLD	0	IN FUNCTIONAL MODE OR DEBUG MONITOR MODE:R/W BY USING MRC/MCR INSTRUCTION (CP14) (ONLY IN SECURE SUPERVISOR MODE) IN DEBUG HALT MODE: NO ACCESS - MCR/MRC INSTRUCTIONS HAVE ANY EFFECT (R/W BY PROGRAMMING THE ICE BY THE JTAG (SCAN 1) IF JSDAEN=1	NO
SECURE TRACE ENABLE BIT	0: ETM IS ENABLED IN NON-SECURE WORLD ONLY. 1: ETM IS ENABLED IN SECURE WORLD AND NON-SECURE WORLD	0	IN FUNCTIONAL MODE OR DEBUG MONITOR MODE:R/W BY USING MRC/MCR INSTRUCTION (CP14) (ONLY IN SECURE SUPERVISOR MODE) IN DEBUG HALT MODE: NO ACCESS - MCR/MRC INSTRUCTIONS HAVE ANY EFFECT (R/W BY PROGRAMMING THE ICE BY THE JTAG (SCAN 1) IF JSDAEN=1	NO
SECURE USER-MODE ENABLE BIT	0: DEBUG IS NOT POSSIBLE IN SECURE USER MODE 1: DEBUG IS POSSIBLE IN SECURE USER MODE	1	IN FUNCTIONAL MODE OR DEBUG MONITOR MODE:R/W BY USING MRC/MCR INSTRUCTION (CP14) (ONLY IN SECURE SUPERVISOR MODE) IN DEBUG HALT MODE: NO ACCESS - MCR/MRC INSTRUCTIONS HAVE ANY EFFECT (R/W BY PROGRAMMING THE ICE BY THE JTAG (SCAN 1) IF JSDAEN=1	NO
SECURE THREAD-AWARE ENABLE BIT	0: DEBUG IS NOT POSSIBLE FOR A PARTICULAR THREAD 1: DEBUG IS POSSIBLE FOR A PARTICULAR THREAD	0	IN FUNCTIONAL MODE OR DEBUG MONITOR MODE:R/W BY USING MRC/MCR INSTRUCTION (CP14) (ONLY IN SECURE SUPERVISOR MODE) IN DEBUG HALT MODE: NO ACCESS - MCR/MRC INSTRUCTIONS HAVE ANY EFFECT (R/W BY PROGRAMMING THE ICE BY THE JTAG (SCAN 1) IF JSDAEN=1	NO

FIG. 61

FUNCTION TABLE

D	CK	Q[n+1]
0	↑	0
1	↑	1
X	↓	Q[n]

LOGIC SYMBOL

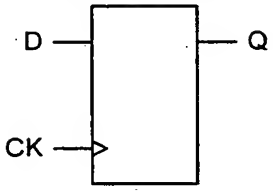

FIG. 62

FUNCTION TABLE

D	SI	SE	CK	Q[n+1]
0	X	0	↑	0
1	X	0	↑	1
X	X	X	↓	Q[n]
X	0	1	↑	0
X	1	1	↑	1

LOGIC SYMBOL

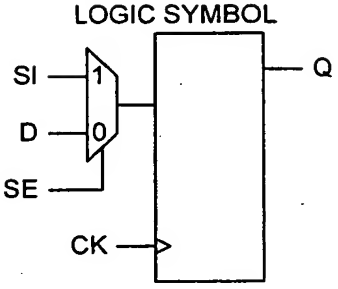

FIG. 63

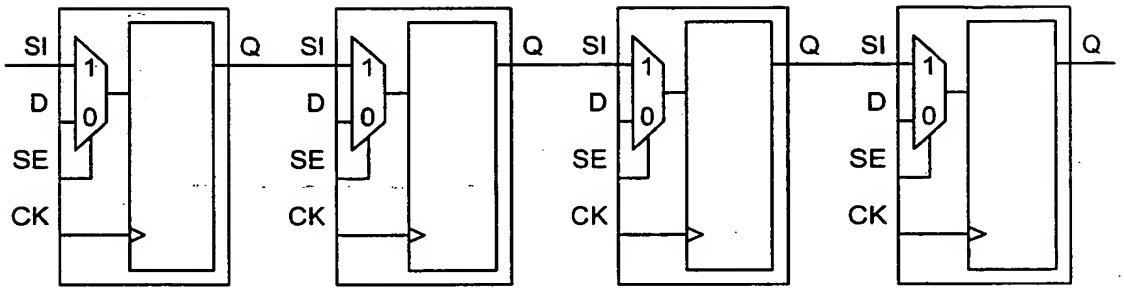
SI
D
SE
CK
1
0
Q
SI
D
SE
CK
1
0
Q
SI
D
SE
CK
1
0
Q
SI
D
SE
CK
1
0
Q

FIG. 64

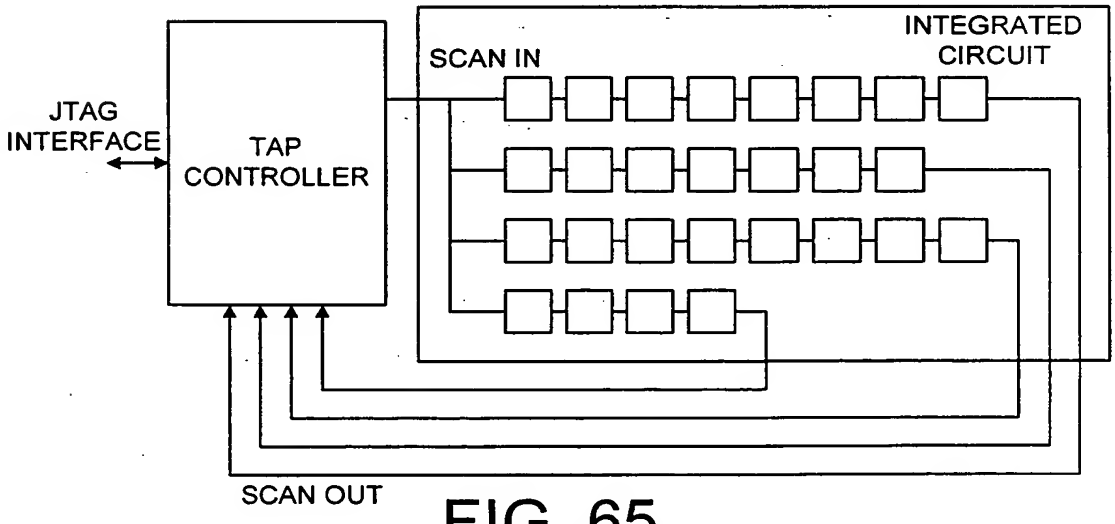
JTAG
INTERFACE
TAP
CONTROLLER
SCAN IN
INTEGRATED
CIRCUIT
SCAN OUT

FIG. 65

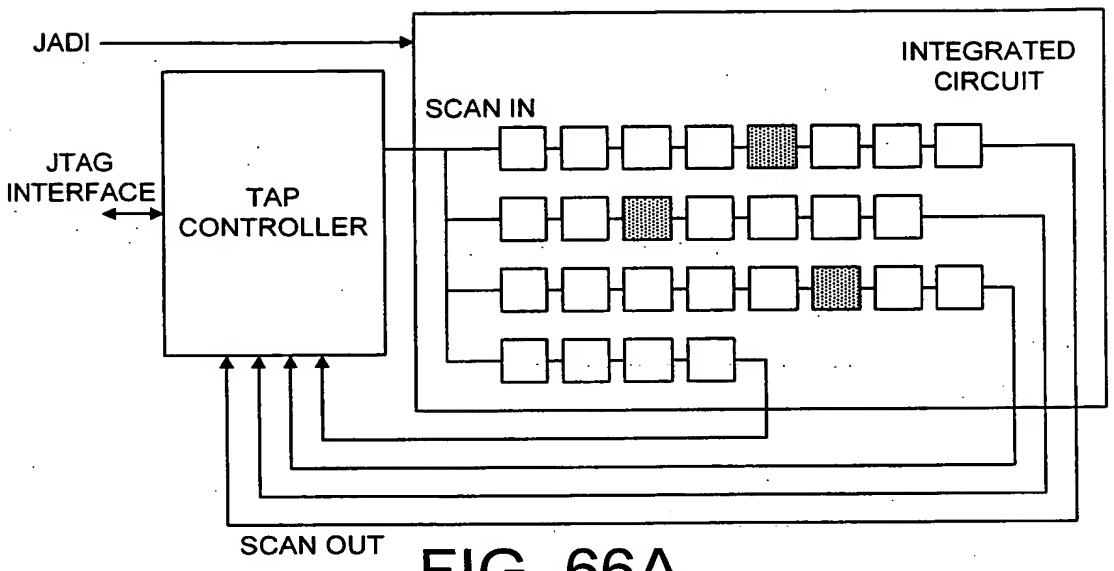

FIG. 66A

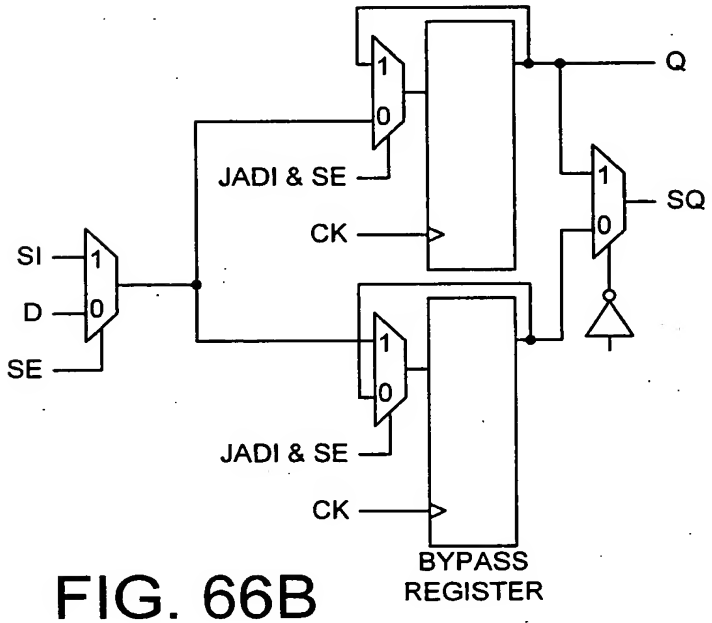

FIG. 66B

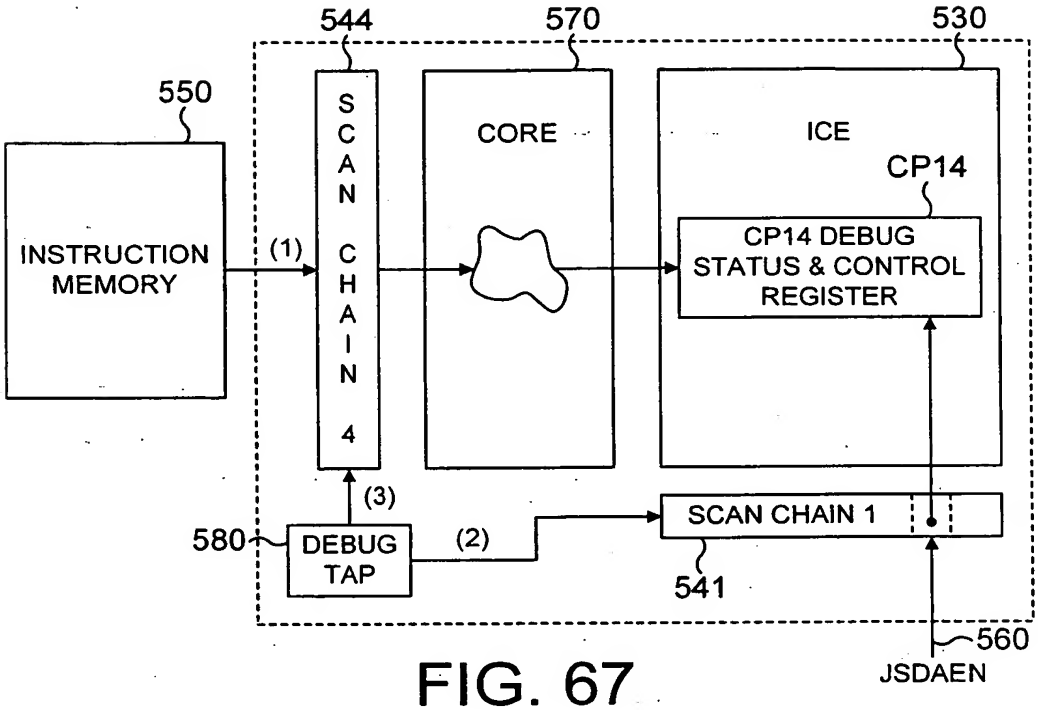
544
570
530
550
SCAN CHAIN 4
CORE
ICE
CP14
INSTRUCTION MEMORY
(1)
CP14 DEBUG STATUS & CONTROL REGISTER
(3)
580
DEBUG TAP
(2)
SCAN CHAIN 1
541
560
JSDAEN

FIG. 67

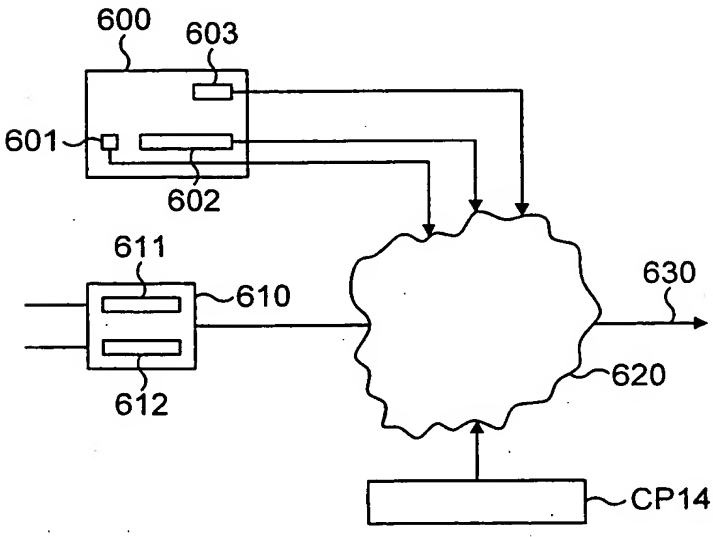
600
603
601
602
611
610
630
620
612
CP14

FIG. 68

CP14 BITS IN DEBUG AND STATUS CONTROL REGISTER			MEANING
SECURE DEBUG ENABLE BIT	SECURE USER-MODE DEBUG ENABLE BIT	SECURE THREAD-AWARE DEBUG ENABLE BIT	
0	X	X	NO INTRUSIVE DEBUG IN ENTIRE WORLD IS POSSIBLE. ANY DEBUG REQUEST, BREAKPOINTS, WATCHPOINTS, AND OTHER MECHANISM TO ENTER DEBUG STATE ARE IGNORED IN ENTIRE SECURE WORLD
1	0	X	DEBUG IN ENTIRE SECURE WORLD IS POSSIBLE
1	1	0	DEBUG IN SECURE USER-MODE ONLY. ANY DEBUG REQUEST, BREAKPOINTS, WATCHPOINTS, AND OTHER MECHANISM TO ENTER DEBUG STATE ARE TAKEN INTO ACCOUNT IN USER MODE ONLY. (BREAKPOINTS AND WATCHPOINTS LINKED OR NOT TO A THREAD ID ARE TAKEN INTO ACCOUNT). ACCESS IN DEBUG IS RESTRICTED TO WHAT SECURE USER CAN HAVE ACCESS TO.
1	1	1	DEBUG IS POSSIBLE ONLY IN SOME PARTICULAR THREADS. IN THAT CASE ONLY THREAD-AWARE BREAKPOINTS AND WATCHPOINTS LINKED TO A THREAD ID ARE TAKEN INTO ACCOUNT TO ENTER DEBUG STATE. EACH THREAD CAN MOREOVER DEBUG ITS OWN CODE, AND ONLY ITS OWN CODE.

FIG. 69A

Inventor: MANSELL et al
SN 10/713,454/Sheet 54 of 56
Atty. Dkt.: 550-485

CP14 BITS IN DEBUG AND STATUS CONTROL REGISTER			MEANING
SECURE TRACE ENABLE BIT	SECURE USER-MODE DEBUG ENABLE BIT	SECURE THREAD-AWARE DEBUG ENABLE BIT	
0	X	X	NO OBSERVABLE DEBUG IN ENTIRE SECURE WORLD IS POSSIBLE. TRACE MODULE (ETM) MUST NOT TRACE INTERNAL CORE ACTIVITY
1	0	X	TRACE IN ENTIRE SECURE WORLD IS POSSIBLE
1	1	0	TRACE IS POSSIBLE WHEN THE CORE IS IN SECURE USER-MODE ONLY
1	1	1	TRACE IS POSSIBLE ONLY WHEN THE CORE IS EXECUTING SOME PARTICULAR THREADS IN SECURE USER MODE. PARTICULAR HARDWARE MUST BE DEDICATED FOR THIS, OR RE-USE BREAKPOINT REGISTER PAIR: CONTEXT ID MATCH MUST ENABLE TRACE INSTEAD OF ENTERING DEBUG STATE

FIG. 69B

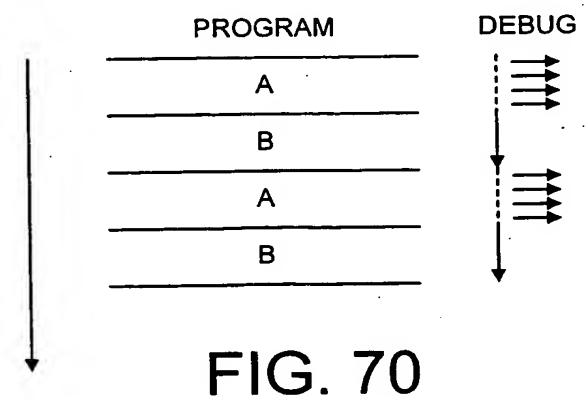

FIG. 70

METHOD OF ENTRY	ENTRY WHEN IN NON-SECURE WORLD	ENTRY WHEN IN SECURE WORLD
BREAKPOINT HITS	NON-SECURE PREFETCH ABORT HANDLER	SECURE PREFETCH ABORT HANDLER
SOFTWARE BREAKPOINT INSTRUCTION	NON-SECURE PREFETCH ABORT HANDLER	SECURE PREFETCH ABORT HANDLER
VECTOR TRAP BREAKPOINT	DISABLED FOR NON-SECURE DATA ABORT AND NON-SECURE PREFETCH ABORT INTERRUPTIONS. FOR OTHER NON-SECURE EXCEPTIONS, PREFETCH ABORT	DISABLED FOR SECURE DATA ABORT AND SECURE PREFETCH ABORT EXCEPTIONS ([1]). FOR OTHER EXCEPTIONS, SECURE PREFETCH ABORT
WATCHPOINT HITS	NON-SECURE DATA ABORT HANDLER	SECURE DATA ABORT HANDLER
INTERNAL DEBUG REQUEST	DEBUG STATE IN HALT MODE	DEBUG STATE IN HALT MODE
EXTERNAL DEBUG REQUEST	DEBUG STATE IN HALT MODE	DEBUG STATE IN HALT MODE

([1]) SEE INFORMATION ON VECTOR TRAP REGISTER

([2]) NOTE THAT WHEN EXTERNAL OR INTERNAL DEBUG REQUEST IS ASSERTED, THE CORE ENTERS HALT MODE AND NOT MONITOR MODE

FIG. 71A

METHOD OF ENTRY	ENTRY IN NON-SECURE WORLD	ENTRY IN SECURE WORLD
BREAKPOINT HITS	NON-SECURE PREFETCH ABORT HANDLER	BREAKPOINT IGNORED
SOFTWARE BREAKPOINT INSTRUCTION	NON-SECURE PREFETCH ABORT HANDLER	INSTRUCTION IGNORED ([1])
VECTOR TRAP BREAKPOINT	DISABLED FOR NON-SECURE DATA ABORT AND NON-SECURE PREFETCH ABORT INTERRUPTIONS. FOR OTHER INTERRUPTION NON-SECURE PREFETCH ABORT	BREAKPOINT IGNORED
WATCHPOINT HITS	NON-SECURE DATA ABORT HANDLER	WATCHPOINT IGNORED
INTERNAL DEBUG REQUEST	DEBUG STATE IN HALT MODE	REQUEST IGNORED
EXTERNAL DEBUG REQUEST	DEBUG STATE IN HALT MODE	REQUEST IGNORED
DEBUG RE-ENTRY FROM SYSTEM SPEED ACCESS	NOT APPLICABLE	NOT APPLICABLE

([1]) AS SUBSTITUTION OF BKPT INSTRUCTION IN SECURE WORLD FROM NON-SECURE WORLD IS NOT POSSIBLE, NON-SECURE ABORT MUST HANDLE THE VIOLATION.

FIG. 71B